A MATRIX APPROACH FOR FINDING EXTREMA: PROBLEMS WITH MODULARITY, HIERARCHY, AND OVERLAP

BY

TIAN-LI YU

B.S., National Taiwan University, 1997
M.S., University of Illinois at Urbana-Champaign, 2003

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2006

Urbana, Illinois

Abstract

Unlike most simple textbook examples, the real world is full with complex systems, and researchers in many different fields are often confronted by problems arising from such systems. Simple heuristics or even enumeration works quite well on small and easy problems; however, to efficiently solve large and difficult problems, proper decomposition according to the complex system is the key. In this research project, investigating and analyzing interactions between components of complex systems shed some light on problem decomposition. By recognizing three bare-bone types of interactions—modularity, hierarchy, and overlap, theories and models are developed to dissect and inspect problem decomposition in the context of genetic algorithms. This dissertation presents a research project to develop a competent optimization method to solve boundedly difficult problems with modularity, hierarchy, and overlap by explicit problem decomposition. The proposed genetic algorithm design utilizes a matrix representation of an interaction graph to analyze and decompose the problem. The results from this thesis should benefit research both technically and scientifically. Technically, this thesis develops an automated dependency structure matrix clustering technique and utilizes it to design a competent black-box problem solver. Scientifically, the explicit interaction model better describes the problem structure and helps researchers gain important insights through the explicitness of the procedure.

To Father and Mother.

Acknowledgments

First of all, I would like to thank my thesis advisor, David E. Goldberg, who gave me this great opportunity to work in the Illinois Genetic Algorithms Laboratory (IlliGAL). Dave is an amazing advisor who always thinks at a level beyond my reach. Our discussions have been inspirative, and his suggestions have shown to be extremely helpful. He encouraged me to raise my bar, and I am glad that I did, considering what I have achieved. I also learned a lot from him about the attitude toward research, and life in general. I feel really lucky to have him as my advisor.

IlliGAL is a great lab with great people. I would like to thank members and visitors of the lab that I had the opportunity to work with; we had great discussions, and they made the lab a fun place to be. Specifically, I want to thank Martin Butz, Jian-Hung Chen, Ying-Ping Chen, Osvaldo Gómez, Alex Kosorukoff, Pier Luca Lanzi, Cláudio Lima, Xavier Llorà, Kei Onishi, Albert Orriols, Gerulf Pedersen, Martin Pelikan, Kumara Sastry, Abhishek Shina, and Noriko Imafuji Yasui. I would like to single out Kumara Sastry, who gave me a lot of help during all these years that I have been in the lab. We had several collaborative publications, which were done through endless discussions, if not arguments, with him.

I also appreciate my committee members: David E. Goldberg, Geneva G. Belford, Sylvian R. Ray, and Ali A. Yassine for agreeing to serve on my committee and their comments to improve this thesis. I also thank Ali Yassine for introducing the dependency structure matrix problem to me, which motivated the work in this thesis.

I would not have been studying in the U.S. from the very beginning if it was not for my parents' and my sister's encouragement and support. I also want to thank my fiancé,

Wan-Lian Chang, who always roots for me and endures my bad temper when I am stressed.

Other than friends in the lab, many friends have also given me different kinds of help to make my life easier and more comfortable in the U.S. I want to thank Chen-Ju Chao, Chien-Yu Chen, Chia-Jung Shih, and Yu-Ping Tseng for their warm friendships.

The work in this thesis was sponsored by the Air Force Office of Scientific Research, Air Force Materiel Command, USAF, under grants F49620-00-0163, F49620-03-1-0129 and FA9550-06-1-0096, the National Science Foundation (NSF) under grants DMI-9908252 and DMR-03-25939, and by the Technology Research, Education, and Commercialization Center (TRECC), at University of Illinois at Urbana-Champaign, administered by the National Center for Supercomputing Applications (NCSA) and funded by the Office of Naval Research under grant N00014-01-1-0175.

The US Government is authorized to reproduce and distribute reprints for government purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the author and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force Office of Scientific Research, the National Science Foundation, the Technology Research, Education, and Commercialization Center, the Office of Naval Research, or the U.S. Government.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

BB	Building block.
BOA	Bayesian optimization algorithm.
DSM	Dependency structure matrix.
DSMGA	Dependency structure matrix genetic algorithms.
ecGA	Extended compact genetic algorithm.
EDA	Estimation of distribution algorithm.
EEL	Effective exchange length.
GA	Genetic algorithm.
GEC	Genetic evolutionary computation.
hBOA	Hierarchical Bayesian optimization algorithm.
hIFF	Hierarchical if-and-only-if problem.
hXOR	Hierarchical exclusive-or problem.
hTrap	Hierarchical trap problem.

List of Symbols

C_b^a	Binomial coefficient.
g	Number of generations.
I	Selection intensity.
k	Order of modules, subfunctions, or building blocks.
l	Problem size.
m	Number of modules, subfunctions, or building blocks.
n	Population size.
s	Selection pressure.
Φ	Cumulative standard Gaussian function.
$\mathbb{H}$	Shannon's entropy.
$\mathbb{I}$	Mutual information.
$\mathbb{W}$	Lambert's W-function.

Introduction

In a variety of different areas, researchers have been investigating complex systems. Complex systems can be both natural and artificial. For example, an atom with complex electric-field interactions between protons and electrons is natural, while an organization with complex communications between employees is artificial. One of the major directions in scientific research is to solve problems in these complex systems. Here the concept of problem solving is very general and includes equation solving, search, optimization, and machine learning. While in the context of genetic algorithm (GA) (Holland, 1975; Goldberg, 1989b), problem solving refers to finding global or near-global optima for a given fitness function. For easy and small-scale problems, simple heuristics or enumeration works quite well. However, to solve the more difficult and larger-scale problems in complex systems effectively and efficiently, one needs to discover the "clouded simplicity" (Simon, 1968) or the "hidden order" (Holland, 1995) of such systems.

Once the complexity (or *complicity*) is better understood, a difficult and large problem can be decomposed into several easier and smaller subproblems. The concept of decomposition can be tracked back at least as far as the 1637 publication of Descartes's *A Discourse on Method* (Descartes, 1994) and lays the foundations of many important problem-solving techniques in computer science, including divide-and-conquer, dynamic programming (Cormen, Leiserson, Rivest, & Stein, 2001), and artificial intelligence planning (Russell & Norvig, 2003).

Suppose we are preparing a working desktop computer system. We need an operating system, application software, and hardware devices. The reason for such decomposition is to

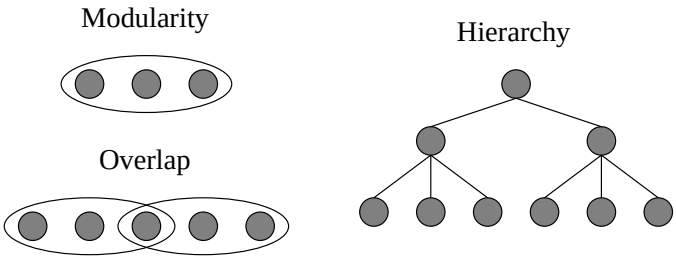


Figure 1: Three different types of interactions: modularity, hierarchy, and overlap.

simplify the task. Each of the subtasks is simpler than the original task, and each subtask should depend less on other subtasks. For example, when using a particular application software, a user should not have to worry about the brand of the underlying hardware.

Solving problems by decomposition is efficient because a proper decomposition minimizes the *interactions* between subproblems. Here the interaction is defined by the problem solver. Two components interact with each other if the problem solver cannot solve the subproblem without the information carried by both components. Based on the similar idea, this research project tries to decompose a complex system by detecting interactions between components of the system. Moreover, this thesis takes a step further to categorize the interactions into three different types: (1) *modularity* (interactions between components), (2) *hierarchy* (interacting components form modules and interacting modules form higher-level modules), and (3) *overlap* (interactions between modules if they share some components) (Figure 1).

Take the computer system in Figure 2 as an example. First of all, the preparation task can be decomposed into hardware and software requirements. Hardware requirements can be further decomposed into the core device requirements, such as processor, motherboard, memory and the peripheral device requirements, like monitor, printer, keyboard, and mouse. Likewise, software requirements can be decomposed into operating system requirements and application requirements. This is the concept of *hierarchy*. Some of these components interact less with some other components. For example, the choice of the application software depends less on the choice of the peripheral devices. Some of these components interact more with some other components. For example, some operating systems only work on a certain

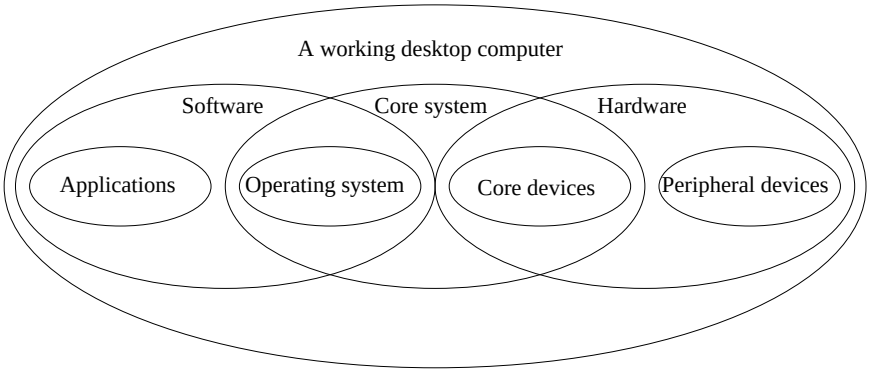


Figure 2: Decomposition of a working desktop computer system.

type of processors, and some application software only works on a certain type of operating systems. Group the operating system and the core devices together, and call it the core system. The type of interactions between the core system and software and that between the core system and hardware is *overlap*. The type of interactions between the components in software, hardware, and the core system is *modularity*.

In this thesis, the idea of decomposing complex systems by detecting interactions between components is applied to the GA field to design a powerful stochastic problem solver that solves difficult problems with modularity, hierarchy, and overlap. Particularly, the work in this thesis develops a technique to extract the interaction information—the information of highly interacting components—by using a dependency structure matrix (DSM) borrowed from the literature of project management and corporate organization. The DSM clustering technique is then utilized to design a competent GA, called the *dependency structure matrix genetic algorithm*, or *DSMGA*.

Thesis Objectives

The objectives of this thesis are twofold. Technically, this thesis presents

1. Designing an automated clustering algorithm that analyzes complex systems by detecting modularity, hierarchy, and overlap.

2. Designing a *competent* optimization algorithm that solves boundedly difficult problems with modularity, hierarchy, and overlap *quickly*, *reliably*, and *accurately* (Goldberg, 2002).

Scientifically, the following ideas play an important role in this thesis.

1. Better understanding the keys to achieve effective recombination for all three types of interactions.

2. Developing useful facetwise models that give insights and guide the design of competent optimization tools.

This thesis develops an automated DSM clustering technique that detects interactions in complex systems. This should benefit system analysis and design such as complex product and organization. In addition, the project designs a competent black-box optimization tool that solves problems via proper decomposition. The optimization tool developed may be used in a variety of applications. First-generation GAs have been widely used both in academics and industry. The method developed here will solve the problems of the first-generation GAs and many others.

Scientifically, the facetwise models developed along the line of the DSMGA++ design help researchers better understand the relationship among interaction-detecting errors, solution quality, and the resulted elongation of the GA run duration. In addition, the DSM provides a powerful visualization tool to describe the dynamics of how the problem is decomposed by the GA. At a higher level, understanding the relationship between modularity, hierarchy, and overlap helps unify the study of complex system, more generally.

Roadmap

This thesis is composed of eight chapters. Chapter 1 gives an introduction to simple GAs, GA design theory, and the complexity of GAs in terms of population size and run duration.

Chapter 2 describes the dependency structure matrix (DSM) and the DSM clustering problem. The chapter proposes a metric to cluster DSMs based on the minimum description length principle. Chapter 3 utilizes the DSM clustering technique to design a competent GA—DSMGA, which solves problems with modularity. Chapter 4 analyzes the scalability of DSMGA by deriving a population-sizing model for entropy-based model-building GAs. It shows that DSMGA solves boundedly difficult problems with modularity within subquadratic number of function evaluations both analytically and empirically. Chapter 5 extends DSMGA to solve problems via hierarchical decomposition. It also demonstrates that the proposed method is able to learn the problem structure. Chapter 6 develops facetwise models to understand how the inaccuracy of the interaction model affects the convergence of GAs. The purpose is to develop useful analytical tools in preparation for the difficulty arising from overlap. Chapter 7 recognizes the keys to conquer the overlap difficulty via investigating the 2D spin-glass problem and designs an effective and efficient recombination method for problems with overlap. Chapter 8 describes the directions of future researches involving the work and concludes this thesis. The remainder of this section describes the content of each chapter in greater detail.

Chapter 1 presents the basic simple GA procedures, the key issues to design a competent GA, and basic facetwise models for the complexity of GAs. This chapter describes the mechanism of several basic GA operators in detail, including truncation and tournament selection, one-point, two-point, and uniform crossover, and bit-wise mutation. It then argues the design of a *competent* GA by introducing the concepts of deception and nearly decomposable problems. The complexity of GAs is discussed from two aspects: population size and run duration. In particular, it gives introductions to the building-block (BB) supply model, the decision-making model, the gambler's ruin model, and the time-to-convergence model. The concepts and models presented in this chapter should provide enough background for readers to follow this thesis.

Chapter 2 introduces the DSM method, the DSM clustering problem, and proposes a clustering metric based on the minimum description length principle (MDL). It starts with a brief survey of existing clustering algorithms, then introduces the DSM method, and discusses the difficulty of DSM clustering. Several criteria for a desirable clustering metric are described. Based on MDL, a clustering metric that satisfies those criteria is proposed. Combined with GAs, the clustering metric is tested on several manually-designed DSMs and a real-world DSM for GM power train development teams. The proposed method is capable of mimicking the clustering preference of human experts by tuning weights of the metric.

Chapter 3 proposes DSMGA, which solves problems via proper problem decomposition. DSMGA utilizes the DSM clustering technique developed in Chapter 2 to analyze the problem structure. Three main tasks—DSM construction, DSM clustering, and BB-wise crossover—are described in detail. Three different interaction-detection metrics—nonlinearity, simultaneity, and entropy—are compared and contrasted in a two-bit scenario. The modularity identification ability of DSMGA is empirically demonstrated.

Chapter 4 analyzes the scalability of DSMGA by proposing a population-sizing model for the entropy-based model building in GAs. The proposed model refines and corrects existing ones. It also preliminarily incorporates the effect of selection pressure and indicates the existence of an optimal selection pressure. The proposed model is empirically verified for ecGA and DSMGA.

Chapter 5 extends DSMGA to another level of optimization by considering hierarchical problems. The chapter first describes several hierarchical problems and the keys to conquer hierarchical difficulty. It then proposes a explicit chunking method—the substructural chromosome compression, which compresses a nearly converged BB by a certain number of the BB's most expressive schemata. The proposed DSMGA+ optimizes problems via hierarchical problem decomposition. When it finishes, the prob-

lem structure is learned and stored in an explicit manner that is comprehensible for human researchers. When tested on several hierarchical problems, DSMGA+ scales sub-quadratically on those problems.

Chapter 6 develops several facetwise models that are useful for investigating problems with overlap. In particular, two types of errors in interaction models are investigated, and their effects on GA convergence are modeled. Based on those facetwise models, this chapter recognizes three keys to achieve effective and efficient recombination: (1) minimization of BB disruptions, (2) maximization of information exchange, and (3) nondeterminism of information exchange.

Chapter 7 investigates how to achieve effective and efficient recombination for problems with overlap. It starts with a simple problem with cyclically overlapping BBs, and then advances to the 2D spin-glass problem, which contains more overlapping BBs. By investigating these problems with overlap, this chapter recognizes three basic guidelines to conquer the overlap difficulty: (1) preservation of alternative solutions, (2) proper sequencing, and (3) well-informed decision. A series of experiments are conducted to verify the essence of these guidelines. The investigation in this chapter also helps explain the success of the hierarchical Bayesian optimization algorithm (hBOA) on the spin-glass problem. The experiments in this chapter demonstrate the use of off-line DSM analysis, and the pros and cons for off-line model building are discussed.

Chapter 8 first describes several possible directions to extend the work presented in this thesis. It discusses several interesting extended topics and their challenges. Finally, it presents important consequences and concludes this thesis.

Chapter 1

Simple GAs, GA Design Theory, and GA Complexity

This chapter gives an introduction to simple genetic algorithms (GAs), GA design theory, and some background of GA complexity. The derivations of the decision-making population-sizing model and time-to-convergence model will be used later in this thesis.

1.1 Introduction to Simple GAs

GAs (Holland, 1975; Goldberg, 1989b) are stochastic search and optimization methods that require only the quality information of solution candidates. The term 'simple GA' refers to a specific class of GAs that does not involve learning problem structures[1]. A simple GA consists of four primary operators: *selection*, *crossover*, *mutation*, and *replacement*. The operand is a population of chromosomes, where each chromosome represents a solution candidate to the given problem. To prepare such a population for a simple GA, two auxiliary operators, *encoding* and *initialization*, are needed. The major input of a simple GA is a *fitness function* that tells the GA the quality of solution candidates. Finally, a termination condition needs to be defined.

The functions of these operators and the execution sequence of a simple GA are described below in detail.

1. **Encoding.** Based on the encoding scheme, a solution candidate is encoded into a chromosome. Depending on different problems and different encoding schemes, a chromosome can be binary, χ-ary, integer, real-valued, and etc. The chromosome

[1]Learning the problem structure is often addressed as *linkage learning* in GA literature.

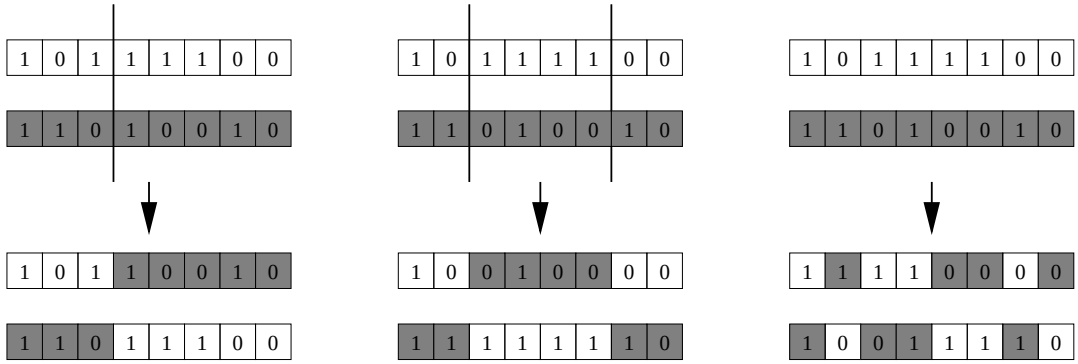
length can also be fixed or variable. This thesis focuses on binary encoding and fixed chromosome length.

2. **Initialization.** At the beginning, the GA creates a population of chromosomes. The alleles of the chromosomes can be randomly initialized or assigned based on some prior knowledge to the problem. The population size n can be fixed or adjustable during the GA run.

3. **Evaluation.** After initialization, the GA applies the fitness function to evaluate the quality of each chromosome in the current population.

4. **Selection.** The objective of selection is to select promising chromosomes that will have their information passed on to the next generation. There are various selection schemes that can be used. In this thesis, *truncation selection* and *tournament selection* are used. With a selection pressure s, truncation selection selects the $\frac{n}{s}$ best chromosomes and puts s copies of them into the *mating pool*. Tournament selection holds tournaments among s randomly selected chromosomes, and put a copy of the best chromosomes into the mating pool. For a population size n, n such tournaments are held to fill up the mating pool. There are two different versions of tournament selection: with or without replacement (Miller & Goldberg, 1995). This thesis adopts tournament selection without replacement, which is less noisy since each chromosome enters the tournament exactly s times.

5. **Crossover.** The crossover operator is performed according to an user-defined probability p_c, usually high, and results in new chromosomes having characteristics taken from the parent chromosomes. This section describes three types of crossover operators: one-point crossover, two-point crossover, and uniform crossover (Figure 1.1). One-point crossover firstly randomly picks a number i between 1 and $l-2$, where l is the chromosome length. It then transfers the information carried by genes x_0 to x_i



(a) One-point crossover. (b) Two-point crossover. (c) Uniform crossover.

Figure 1.1: One-point, two-point, and uniform crossover.

from a parent to one child and the information carried by genes x_{i+1} to x_{l-1} to the other child. Similarly, two-point crossover randomly picks two distinct numbers i and j between 0 and $l-2$ as cross sites, where $i < j$. A child then inherits alleles x_{i+1} to x_j from one parent and the rest from the other parent. The uniform crossover operator randomly switches each gene of the two parent chromosomes with a certain probability (usually 0.5) to produce two new offspring chromosomes. These crossover operators have different biases and mixing abilities. For instance, the alleles of the first gene x_0 and the last gene x_{l-1} will always be exchanged in one-point crossover. In two-point crossover, the alleles of adjacent genes are more likely to be transferred together. In uniform crossover, every gene has an equal probability (0.5) to be transferred with other genes. For more detailed information on the difference of these crossover operators, readers are referred to Sastry and Goldberg (2002).

6. **Mutation.** The mutation operator performs according to an user-defined probability p_m, usually low, and serves to introduce some variability into the gene pool. For a binary chromosome, the bit-wise mutation inverts the value of genes with the mutation probability p_m. Without mutation, offspring chromosomes would be limited to only the genes available within the initial population.

7. **Replacement.** The simple full replacement replaces the whole parent population

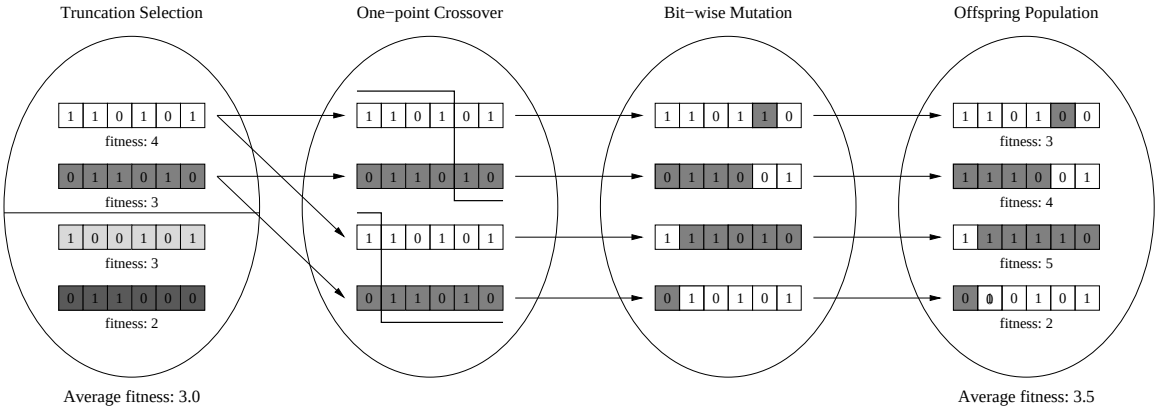


Figure 1.2: The simulation of one generation of a simple GA. The simple GA adopts truncation selection, one-point crossover, and simple bit-wise mutation. The fitness values are calculated by OneMax. As indicated in the figure, the average fitness increased after one generation.

with the newly generated offspring population. *Elitism* (Goldberg, 1989b) can be also adopted here. For example, keep 50% of the best parent chromosome and replace the other 50% with the newly generated chromosomes.

8. **Repeat steps 3 through 7 until the termination condition has been met.** The termination condition can be a simple criterion based on the number of generations, the number of function evaluations, or the fitness convergence, to name a few. It can also be a combination of several simply criteria.

Figure 1.2 shows the simulation of one generation of a simple GA with truncation selection, one-point crossover, bit-wise mutation, and full replacement. The problem is a OneMax problem where the fitness is defined as

$$f(\vec{x}) = \Sigma_i x_i, \quad x_i \in \{0, 1\}. \tag{1.1}$$

The global optimum is a binary string with all ones. As the figure shows, the average fitness of the population is higher than that of the previous generation.

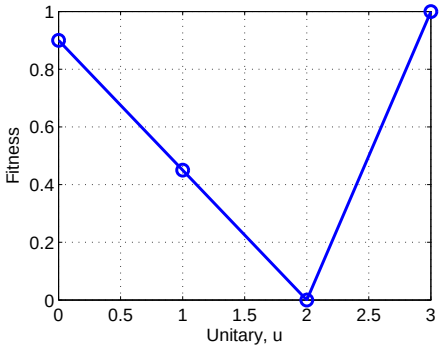


Figure 1.3: A 3-bit trap function. The design of the trap function deceives any hill-climber to the wrong answer—000.

1.2 GA Design Theory

Simple GAs (Goldberg, 1989b) have been widely used in many different fields for optimization. However, simple GAs only utilize the information of one single bit, and that can be misleading. The fact that a single bit can be misleading may be best described by the *trap* function (Goldberg, 1987). A 3-bit trap is given by

$$f^3_{trap} = \begin{cases} 0.9 & \text{if } u = 0 \\ 0.45 & \text{if } u = 1 \\ 0 & \text{if } u = 2 \\ 1.0 & \text{if } u = 3, \end{cases} \tag{1.2}$$

where u is the number of 1's (Figure 1.3). Suppose that the problem is composed of several such trap functions and that the neighbor is randomly assigned. For example, $f(\vec{x}) = f^3_{trap}(x_1, x_7, x_{29}) + f^3_{trap}(x_2, x_9, x_{13}) + f^3_{trap}(x_3, x_{14}, x_{18}) + \cdots$. A simple GA, which does not capture the problem structure, would tend to give the solution containing all 0's.

Simple GAs assume that highly interacting genes, if any, are encoded next to each other so that the crossover operator does not disrupt *building blocks* (BBs) (Goldberg, 2002), where a BB is a module—a group of highly interacting genes. Simple GAs would work well on such

problems. However, the assumption is not true for the above trap example and for many real-world problems. It has been shown elsewhere that for some problems, interactions need to be handled carefully to achieve satisfactory solutions (Santarelli, Yu, Goldberg, Altshuler, O'Donnell, Southall, & Mailloux, 2006).

Competent GAs (Goldberg, 2002) are GAs with interaction-detection techniques designed to decompose boundedly difficult problems in order to find global, or near global solutions within a sub-quadratic number of function evaluations. To better understand the concept of boundedly difficult problems, consider two extreme cases: the OneMax problem and the needle-in-a-haystack (NIAH) problem. In the OneMax problem, the fitness value of a binary chromosome is defined as the number of ones in the chromosome, and hence the chromosome with the highest fitness value would be of all ones. In NIAH, the fitness value for one particular chromosome is highest (*e.g.*, 1) while the fitness values for all other chromosomes are equally low (*e.g.*, 0). At one extreme, the genes in the OneMax problem are independent with respect to each other, and hence the OneMax problem is fully decomposable. OneMax is considered GA-easy, and a simple GA can easily solve the problem within a sub-quadratic number of function evaluations. At the other extreme, the order of interaction in the NIAH problem is equal to the chromosome length, and hence the NIAH problem is not decomposable. The NIAH problem is considered to be GA-difficult and has been shown that no algorithm can do any better than a random search for this type of problem. On average, an exponentially large number of function evaluations are required to find the optimal solution to the NIAH problem. The OneMax and NIAH problems are either too simple or too difficult to be interesting. Nevertheless, can we efficiently solve a problem that falls somewhere in between these two extremes? For example, consider an additive NIAH problem, which is merely a concatenation of several order-bounded NIAHs. Can GAs find the optimum quickly? More generally, can we design GAs that solve *nearly decomposable* problems *quickly*, *reliably*, and *accurately* (Goldberg, 2002)?

Starting from Holland's notion (Holland, 1975) of a BB, a decomposition methodology

has been proposed for the successful design of competent GAs (Goldberg, Deb, & Clark, 1992; Goldberg, 2002):

1. Know what GAs process—BBs.
2. Know thy BB challengers—BB-wise difficult problems.
3. Ensure an adequate supply of raw BBs.
4. Ensure increased market share for superior BBs.
5. Know BB takeover and convergence times.
6. Make decisions well among competing BBs.
7. Mix BBs well.

Much work has been done investigating each of these critical categories, including problem difficulty (Goldberg, 1989a; Deb & Goldberg, 1993), adequate supply (Goldberg, 1989c; Goldberg, Sastry, & Latoza, 2001a), decision making (Goldberg, Deb, & Clark, 1992; Harik, Cantú-Paz, Goldberg, & Miller, 1997), and mixing (Thierens & Goldberg, 1993; Sastry & Goldberg, 2002). Of all the categories mentioned above, effective mixing has been found to be one of the most essential and challenging issues for GA success, and efforts have been put forth in developing competent GAs to address this issue. A number of competent GAs are described in Goldberg (2002) and Larrañaga and Lozano (2002).

1.3 Population-Sizing Models

Facetwise and dimensional models have been very effective not only in the design of genetic algorithms, but also in understanding GA dynamics and mechanisms. This section briefly outlines the models dictated by BB supply and decision making. This section emphasizes on the derivations of the decision-making model since a population-sizing model based on decision making will be presented later in this thesis.

1.3.1 BB supply model

The first step towards understanding population sizing is to tackle the issue of BB supply, where the minimum population size required to ensure the presence of at least one copy of all raw schemata is modeled. Holland (1975) estimated the number of BBs that receive at least a specified number of trials using Poisson distribution. A later study (Goldberg, 1989b) calculated the same quantity more accurately using binomial distribution and studied their effects on population sizing in serial and parallel computations. Reeves (1993) proposed a population sizing model for supply of alphabets with fixed cardinality. Recently, Goldberg, Sastry, and Latoza (2001b) developed facetwise models for ensuring BB supply in the initial population for genetic algorithms. They considered a population of fixed-length strings consisting alphabets of cardinality χ and predicted that the population size required to ensure the presence of all competing BBs with a tolerance of $\epsilon = 1/m$ is given by

$$n = \chi^k \left(k \log \chi + \log m\right), \tag{1.3}$$

where k is the order of BBs, and m is the number of BBs.

1.3.2 Decision-making population-sizing model

Goldberg, Deb, and Clark (1992) proposed a population-sizing model based on decision making. The basic idea is that the population size should be large enough to make correct decision between the correct BB and the most competing incorrect BB. Since the decision-making model is later utilized in this thesis, this subsection shows some details of the derivations.

Define two schemata H_0 and H_1, where H_0 is the most competing incorrect BB and H_1 is the correct BB. The fitness difference between H_0 and H_1 is called the *minimal signal*:

$$d_{min} = f_{H_1} - f_{H_0}, \tag{1.4}$$

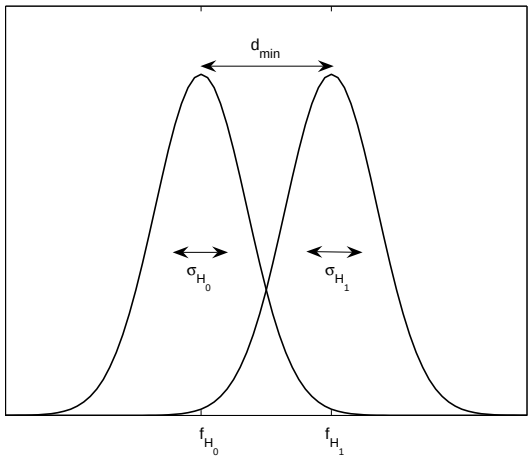


Figure 1.4: The decision making between two competing BBs.

where f_{H_1} and f_{H_0} are the fitness values of H_1 and H_0 respectively. If BBs are uniformly scaled, the fitness variances of H_0 and H_1 are both approximately $(m-1)\sigma_{BB}^2$, where m is the number of BBs in the problem and σ_{BB}^2 is the fitness variance of one BB.

Given a BB of order k, there are total of 2^k competing schemata for that BB. For a GA with a population of size n, initially H_0 and H_1 should both contain roughly $\frac{n}{2^k}$ chromosomes. If n is large enough, which is usually the case in GAs, the distribution of the sampled fitness can be approximated as a Gaussian distribution according to the central limit theorem (Feller, 1966) (Figure 1.4). The variances of the sampled fitness of H_0 and H_1 can then be approximated as:

$$\hat{\sigma}_{H_0}^2 = \hat{\sigma}_{H_1}^2 = (m-1)\sigma_{BB}^2 \frac{2^k}{n}. \tag{1.5}$$

Define a decision variable,

$$\tau = \frac{f_{H_1} - f_{H_0}}{\sqrt{\hat{\sigma}_{H_0}^2 + \hat{\sigma}_{H_1}^2}}. \tag{1.6}$$

For a problem with m BBs, if at most one BB is allowed to be incorrect, the following relationship holds: $\Phi(\tau) \geq 1 - \frac{1}{m}$, where Φ is the cumulative standard Gaussian distribution.

With arithmetic manipulation, the inequality can be approximated as

$$n \geq c2^k m \log m \frac{\sigma_{BB}^2}{d_{min}^2}, \tag{1.7}$$

where c is a problem dependent constant.

1.3.3 Gambler's ruin population-sizing model

The decision-making model incorporates noises arising from other partitions. However, it assumes that if an incorrect decision between competing BB is made in the first generation, GAs are unable to recover from that error. Harik, Cantú-Paz, Goldberg, and Miller (1997) refined the decision-making model by incorporating cumulative effects of decision making over time rather than in the first generation only. They modeled the decision making between the correct and the most competing incorrect BBs in a partition as a gambler's ruin problem. An approximated form of the population-sizing model is given by (Harik et al., 1997):

$$n = \frac{\sqrt{\pi}}{2} \frac{\sigma_{BB}}{d} 2^k \sqrt{m} \log m, \tag{1.8}$$

where k is the BB size, m is the number of BBs, d is the minimal signal between competing BBs, and σ_{BB} is the fitness variance of one BB. The above equation assumes a failure probability, $\alpha = 1/m$.

1.4 Convergence Time Model

To analyze the complexity of GAs, models for GA convergence time is also required. Mühlenbein and Schlierkamp-Voosen (1993) defined the *selection intensity* for GAs on a maximization problem as:

$$I = \frac{\bar{f}_{t+1} - \bar{f}_t}{\sigma_t}, \tag{1.9}$$

where I is the selection intensity, $\bar{f}_t$ is the average fitness of the population at generation t, and σ_t is the standard deviation of the fitness of the population at generation t. For the tournament selection with a fixed tournament size, the selection intensity is a constant (Blickle & Thiele, 1995).

In the OneMax problem, every bit is independent, and the fitness is of binomial distribution. Define b_t as the proportion of ones in the population at generation t. It is easily seen that $f_t = l \cdot b_t$ and $\sigma_t = \sqrt{l \cdot b_t(1 - b_t)}$. The time-to-convergence model can be derived from the following equation (Mühlenbein & Schlierkamp-Voosen, 1993; Thierens & Goldberg, 1994).

$$b_{t+1} - b_t = \frac{I}{\sqrt{l}} \sqrt{b_t(1 - b_t)}. \tag{1.10}$$

Miller (1997) extended the time-to-convergence model to problems with uniformly scaled BBs. If the interaction model successfully identifies every BB, by treating correct BBs as 1's and incorrect BBs as 0's, the problem is mapped to the OneMax problem. The only difference is that the growth of correct BBs is usually slower. The reason is that a chromosome with more correct BBs does not always have a higher fitness value. For example, {0" 0" 1} might have a lower fitness value than {0' 0' 0'}, where 1 represents a correct BB, 0' represents an incorrect BB with a high fitness value, and 0" represents another incorrect BB with a low fitness value. The growth of correct BBs is modeled as follows.

$$p_{t+1} - p_t = \frac{I'}{\sqrt{m}} \sqrt{p_t(1 - p_t)}, \tag{1.11}$$

where I' is the BB-wise selection intensity, and p_t is the proportion of correct BBs at generation t.

When p_t increases slowly, the above difference equation can be approximated as a differential equation:

$$\frac{dp(t)}{dt} = \frac{I'}{\sqrt{m}} \sqrt{p(t)(1 - p(t))}, \tag{1.12}$$

or

$$\frac{dp(t)}{\sqrt{p(t)(1-p(t))}} = \frac{I'}{\sqrt{m}}dt. \tag{1.13}$$

Equation 1.13 can be solved by taking integration on both sides, which yields

$$\sin^{-1}(2p(t)-1) - \sin^{-1}(2p(0)-1) = \frac{I'}{\sqrt{m}}t. \tag{1.14}$$

Solving the above equation for $p(t_{conv}) = 1$ yields the convergence time for GAs.

$$t_{conv} = \left(\frac{\pi}{2} - \sin^{-1}(2p(0)-1)\right)\frac{\sqrt{m}}{I'}. \tag{1.15}$$

Given the order of BBs is k, $p(0) \simeq \frac{1}{2^k}$, which is usually small for most problems. By neglecting the proportion of correct BBs at the beginning of the GA run, the time-to-convergence model can be further simplified as

$$t_{conv} \simeq \frac{\pi\sqrt{m}}{2I'}. \tag{1.16}$$

1.5 Summary and Conclusions

This chapter first gives an introduction to simple GAs, followed by motivation to problem decomposition in GAs. It then gives some background of GA complexity. Based on the facetwise models of population-sizing and run duration, given a perfect problem structure information, a GA should consume $\Theta(m \log m)$ to $\Theta(m^{1.5} \log m)$ number of function evaluations. In other words, a properly designed GA should solve boundedly difficulty problems in sub-quadratic time given the problem structure is known. Nevertheless, in most real-world problems, the problem structure information does not come for free. The rest of this thesis focuses on how to learn problem structures and how to achieve efficient and effectively recombination for problems with modularity, hierarchy, and overlap.

Chapter 2

Analyzing Complex Systems via DSM Clustering

This chapter describes how to analyze complex systems by dependency structure matrix (DSM) clustering techniques. A DSM is a matrix that contains the information of pair-wise interaction between every pair of the components in a system. The objective of DSM clustering is to transfer the pair-wise interaction information into higher-order interaction information. DSM clustering techniques are known to be extremely useful in architectural improvement in organizations and product design and development (McCord & Eppinger, 1993; Pimmler & Eppinger, 1994). The clustering techniques developed in this chapter will be used later in this thesis as a module detector to design an advanced optimization algorithm.

This chapter is organized as follows. Starting from a brief review of existing clustering methods from graph theory and engineering design literature, it then gives a brief introduction to the DSM method, the terminology, and the complexity of DSM clustering. Subsequently, a clustering metric is proposed based on the minimum description length principle. Combining with a simple genetic algorithm, the clustering metric is tested on several manually designed DSMs. Finally, the proposed clustering method is demonstrated on an engine product development team at GM, and comparison between the proposed method and manual clustering concludes this chapter.

2.1 Literature Review

Formally, a clustering of a graph $G = (N, E)$, where N is a set of components and E is a set of weighted edges, is a partition of N into disjoint subsets (Hartigan, 1975), called *modules*. Many constraints can be specified to limit the size of each module or the total number of modules, among others. Every non-trivial variation of the graph clustering problem is known to be NP-hard (Garey, Johnson, & Stockmeyer, 1976), and therefore, numerous heuristics have been used to provide approximated but fast solutions. These heuristics vary depending on the specific application and objective function chosen (Anderberg, 1973), which include soft computing methods (Kirkpatric, Gelatt, & Vecchi, 1983; Glover, 1989; Glover, 1990; Bui & Moon, 1996), spectral methods (Gaertler, 2002), and operations research methods (Hansen & Jaumard, 1997). A lengthy review on this large stream of graph theory literature is outside the scope of this thesis; however, the reader is referred to (Jian, Murty, & Flynn, 1999) for a comprehensive review.

Tools for identifying product modules are scarce in the engineering design literature; however, the few ones found are limited to guidelines and heuristics (Alexander, 1964; Rechtin & Maier, 1997). Early clustering algorithms for engineering design and underlying principles are described in (Alexander, 1964). The relevant clustering metric proposed is some function of the number of linkages that cross a partition boundary. In the work of Alexander (1964), the objective function was implemented with a simple hill climbing search strategy. More recently, Stone, Wood, and Crawford (2000) proposed the use of *function diagrams* for identifying product modules. This technique starts with creating a function structure for the product under consideration, followed by grouping the sub-functions into modular chunks. Other techniques such as Hatley/Pirbhai method (Zakarian & Rushton, 2001) and interaction graphs (Kusiak & Huang, 1996) have also been used for the development of modular products. All these methods rely on undirected search for modules in the structure and are likely to result in non-optimal and non-unique modular architectures. Furthermore,

these methods rely on mapping the functional decomposition of the product to the physical architecture. A module in this way becomes the physical realization of a function, and the whole interface problem is not properly addressed. On the other hand, this chapter does not address itself to the function/form relationship and relies merely on the physical components of the product and the interactions among them.

In addition to graph theoretic metrics, Wang and Antonsson (2004) proposed an information theoretic metric for detecting modularity. The metric was used in conjunction with a genetic algorithm to search for optimal modular configurations. The method in this chapter differs from their work by the capability of detecting overlapping modules and bus structures in the product architecture and the capability of mimicking human experts' preference. Another clustering metric was proposed by Baldwin and Clark (2000) using real options theory where they calculated the net options value of modularity for various product architectures. Sharman, Yassine, and Carlile (2002) have experimented with such a clustering metric and reported many difficulties when dealing with complex product architectures that do not possess a pure hierarchical structure.

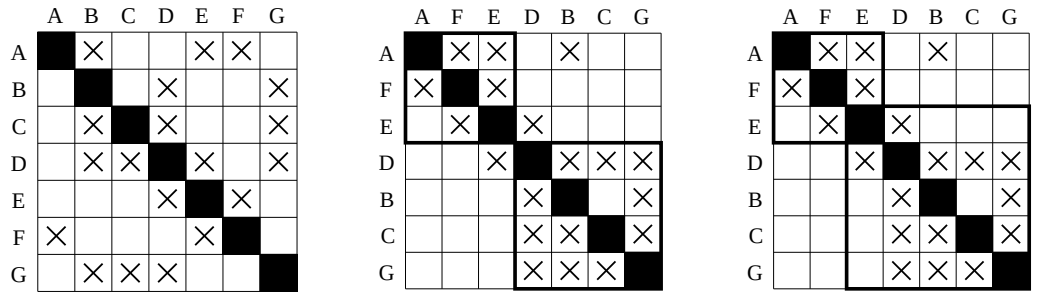
Another research direction related to the work in this chapter focuses on the development of product families based on modular product platforms that allow sharing of core modules among different products (Meyer & Lehnerd, 1997; Robertson & Ulrich, 1998). There are several approaches for designing different kinds of product platforms. Gonzalez-Zugasti, Otto, and Baker (2000) formulated the design of a platform-based product family as an optimization problem in which the advantages of designing a common platform must be balanced against the constraints of the individual product variants. Their approach allows identifying modules that could be made common to several product variants. Simpson, Maier, and Mistree (2001) used a decision support problem formulation to design families of products based on scalable platforms. Martin and Ishii (2002) proposed two indices to measure a product architecture, which were used by design teams to develop a decoupled architecture that requires less design effort for follow-on products. The work in this chapter

differs from this research stream in that it investigates the "optimal" module architecture for a single product without any commonality considerations that arise in family approaches. However, this chapter complements the platform development literature by providing a pre-processing phase to identify core platform modules common to all variants, and individual modules that create differentiation among family members.

2.2 Dependency Structure Matrix (DSM) Method

A DSM is a matrix representation of a graph. The vertices of the graph, representing the components in a complex system, correspond to the column and row headings in the matrix (Eppinger, Whitney, Smith, & Gebala, 1994; Yassine, Jogleker, Braha, Eppingher, & Whitney, 2003). The arrows, representing relationships between components, correspond to the '×' marks inside the matrix. For example, if there is an arrow from vertex C to vertex A, then a '×' mark is placed in row A and column C. Diagonal entries have no significance and are blacked out in the following figures. Alternatively, one can use '1' and '0' to replace '×' and blank, respectively, making the DSM a binary matrix $[d_{ij}]$ with entries $d_{ij} = 0$ or 1 for $i \neq j$ and $d_{ii} = 1$ by assumption.

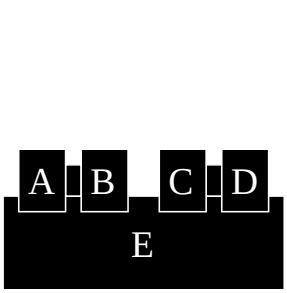
Once the DSM for a product is constructed, it can be analyzed for identifying modules, a process referred as clustering. The goal of DSM clustering is to find a clustering arrangement where modules minimally interact with each other while components within a module maximally interact with each other (Fernandez, 1998). As an example, consider the DSM shown in Figure 2.1(a). One can see from Figure 2.1(b) that the original DSM was rearranged by permuting rows and columns to contain most of the interactions within two separate modules: $\{A, F, E\}$ and $\{D, B, C, G\}$. However, three interactions are left out of any modules. An alterative arrangement is suggested in Figure 2.1(c). This arrangement suggests the forming of two overlapping modules: $\{A, F, E\}$ and $\{E, D, B, C, G\}$. It eliminates two left-out interactions by introducing sparser modules.



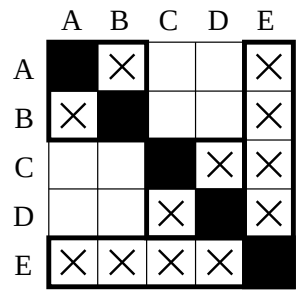
(a) Original DSM. (b) Clustered DSM. (c) Alternative clustering.

Figure 2.1: DSM clustering examples.

The DSM representation of a system/product architecture has shown useful because of the visual appeal and simplicity. Numerous researchers have used it to propose architectural improvements by manipulating the order of rows and/or columns in the matrix (McCord & Eppinger, 1993; Pimmler & Eppinger, 1994). In an attempt to automate this manual process of DSM inspection and manipulation, Fernandez (1998) used the simulated annealing search technique to find "good" DSM clustering arrangements. In his approach, each component starts out by being an individual module and evaluates bids from all the other modules. If any module is able to make a bid that is better than the current base case then the component is moved inside the module. The objective function is therefore a tradeoff between the cost of being inside a module and the overall system benefit. Sharman, Yassine, and Carlile (2002) attempted using Fernandez's algorithm on an industrial gas turbine. However, they showed that this algorithm is incapable of predicting the formation of "good" clustering arrangements for complex product architectures due to the oversimplification of the objective function and the frequent susceptibility of the search algorithm used to be trapped in local optima. In a similar venue, Whitfield, Smith, and Duffy (2002) used GAs to form product modules. Their algorithm is also built upon Fernandez's concept and suffers from similar problems consequently.



(a) Physical Schematic.



(b) The Corresponding DSM.

Figure 2.2: The physical schematic and the corresponding DSM of a simple product. The product is composed by two modules and one simple bus.

2.3 DSM Building Blocks: Product Architecture Terminology

This section defines the basic syntax and semantics for analyzing DSMs in order to characterize the architecture of a complex product and identify its modules. The analysis proceeds using simplified DSMs and product architectures. These simple constructs can be used as building blocks for analyzing more complex architectures (Sharman & Yassine, 2004).

Consider the product architecture depicted by the physical schematic in Figure 2.2(a). It shows five interacting *components*, which are defined as the smallest units in a system decomposition. The situation may be visualized as component E being some form of system-level integrating component, called a *bus*, which connects all other components A, B, C, and D. In this instance, the relationships between the components are symmetric. Component A depends on component B in the same way that B depends on A. The pair of components A and B forms a *module* AB and is related symmetrically to component E. A similar structure can be seen in the pair of components C and D. In the DSM, this pairing is defined as a module. Figure 2.2(b) draws the corresponding DSM of the system.

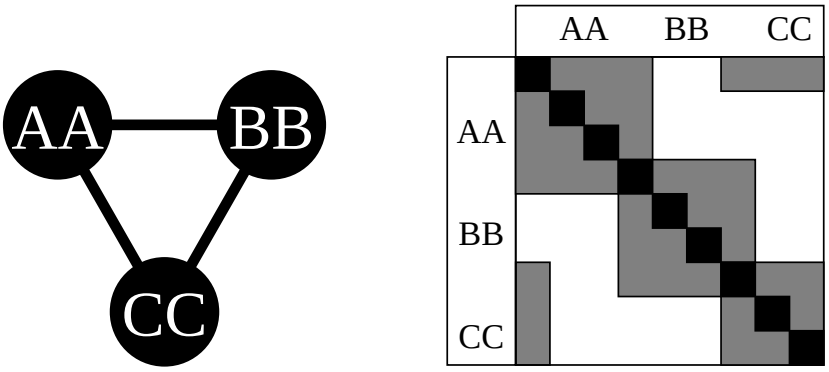


Figure 2.3: Planar triangular modules. One module has to be broken in the DSM representation.

2.4 DSM Clustering Complexities

The challenge of applying automated clustering algorithms to complex DSMs comes from the difficulty of extracting the relevant information and then conveying the information to users. This is most noticeable in the poor handling of the path-dependent situations and modules with a three-dimensional topology.

2.4.1 Path dependency in clustering

Consider a triangular arrangement with symmetrical relationships that can be loosely clustered into three similar modules AA, BB, and CC, as shown in Figure 2.3. The DSM for this physical arrangement can only show two modules and must break up the third. The way the clustering algorithm operates will be path dependent since once the algorithm has started to cluster on any two components, it is unlikely to reverse back to explore a different configuration. This situation may occur depending on how the clustering algorithm perceives the raw data. For example, branch and bound algorithms that are presented with a partially clustered starting point may never branch widely enough to evaluate alternative solutions. For the example in Figure 2.3, module CC has been broken up even though it is identical to the other two clusters in all aspects. Module AA or BB could be broken up equally likely when positioned where CC is.

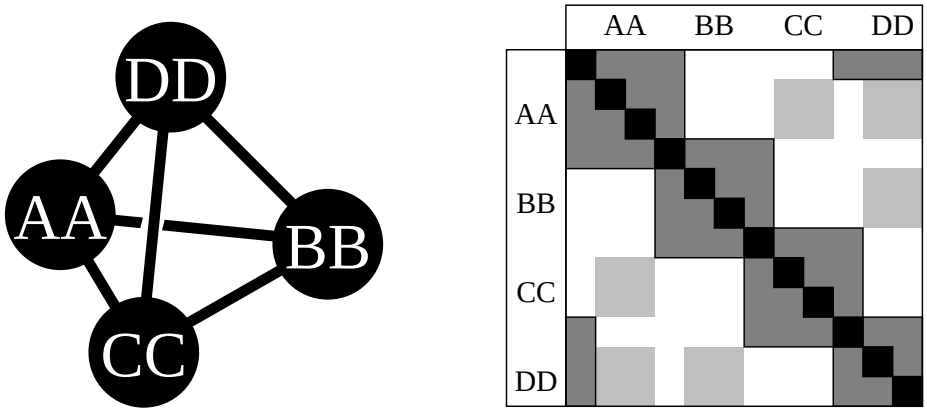


Figure 2.4: Tetrahedron of modules. In the 2D DSM representation, many modules have to be broken. The overlaps between modules (light-grayed areas) make DSM difficult for human to understand or cluster.

2.4.2 Dimensions and topology

Consider a simple three-dimensional structure such as a tetrahedron. This is depicted in Figure 2.4 showing four equal modules, each with dense internal interactions and weak external interactions. If all the modules are perfectly equal, it is purely a matter of chance how any clustering algorithm would present a clustering arrangement. In this example, module DD is the one visually disrupted most by being presented last in the sequence. This results in the effect of spreading its inter-module interactions over a wider spatial area, depicted in the DSM in light gray. To an untrained observer, this might be thought as a bus where module DD is the unique possessor of system-wise integrating functions and some random cross-links in the zone AA-CC.

2.5 MDL-based DSM Clustering Metric

As mentioned earlier, existing DSM clustering algorithms can be found elsewhere (McCord & Eppinger, 1993; Fernandez, 1998; Thebeau, 2001; Whitfield, Smith, & Duffy, 2002). Experiments with these algorithms have shown that those clustering metrics are insufficient for accurately predicting "good" clustering arrangements (Sharman, Yassine, & Carlile, 2002;

Sharman & Yassine, 2004). The lack of an effective clustering method motivates the idea of developing an information theoretic clustering metric. The previous section outlines the requirements necessary for the development of the new clustering metric and the corresponding algorithm:

1. Suggesting an appropriate total number of modules.

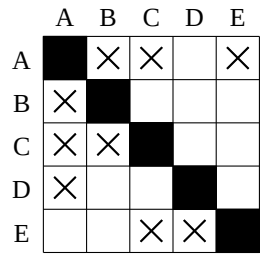
2. Detecting the existence of bus modules.

3. Detecting overlapping modules and modules with a three dimensional structure.

While the first two requirements can be addressed by an appropriate clustering metric, the third requirement is directly related to the encoding and search strategy, discussed in Chapter 2.6.

2.5.1 The minimum description length principle (MDL)

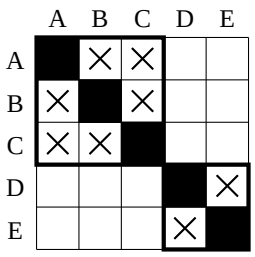
Suppose that a model describing a given data set is presented. Usually, the model does not completely describe the given data due to complexity issues. Therefore, the description of the given data consists of two parts: *model description* and *mismatched data description*. This scheme is easier to understand in light of the following sender-receiver example. Assume a sender has a given data set needed by a receiver. Given a model that approximately describes the given data set of the DSM in Figure 2.5(a), the sender first sends the model description as in Figure 2.5(b) to the receiver. The model description is a hypothetical description of the data set; it always assumes that components within a module maximally interact with each other and that modules do not interact with each other at all. The model description simply provides the number of modules and the names of components contained in each module. For the data set in Figure 2.5(a), the sender would send the following: "**Module 1:** A**,** B**,** C**; Module 2:** D**,** E." Then, the receiver understands the data as in Figure 2.5(b). Note that the hypothetical model presented in Figure 2.5(b) does not completely represent the



(a) The given data set.

Cluster 1: A, B, C
Cluster 2: D, E

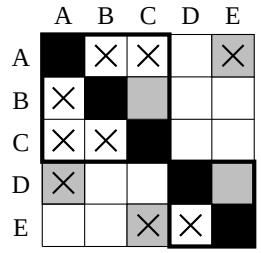
(b) The model description.



(c) The DSM created based on the model description.

	Hypothetical marks that do not exist in the real data set	Real marks that the hypothetical model misses
Inside module	Row B, Column C Row D, Column E	ϕ (always)
Outside module	ϕ (always)	Row A, Column E Row D, Column A Row E, Column C

(d) The mismatched data description.



(e) Combining the model description and the mismatched data description (the grey cells), the original data set is recreated.

Figure 2.5: Obtaining the original data set from the model description and the mismatched data description.

given data set. To ensure that the receiver gets exactly the same data, the sender also needs to send the correct data that were mis-described by the model, as seen in Figure 2.5(c).

Note that if a model is simple, its description is short; however, many data mismatches would take place, resulting in a longer mismatched data description. On the other hand, a complicated model reduces the description of mismatched data while increases the model description. The minimum description length principle (MDL) (Rissanen, 1978; Rissanen, 1999; Barron, Rissenen, & Yu, 1998; Lutz, 2002) satisfies the need for dealing with the above tradeoff. MDL can be interpreted as follows. Among all possible models, choose the model that uses the minimal total length for both model description and mismatched data description. In other words, MDL tries to find a simple model that well describes the given data. Two key points should be noted when using MDL: (1) the encoding should be uniquely

decodable, and (2) the length of encoding should reflect the complexity. For example, the encoding of a complicated model should be longer than that of a simple model.

2.5.2 Model description

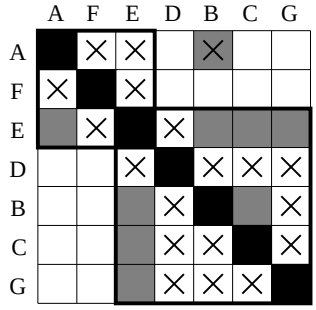
This chapter encodes a model in a straightforward manner. The description of each module starts with a number that is sequentially assigned to each module, followed by a sequence of components in the module. Figure 2.6 shows the corresponding model description of the DSM clustering arrangement in Figure 2.1(c). As the figure indicates, the model description can be calculated as follows.

$$\Sigma_{i=1}^{n_M} \left(\log n_c + M_i \cdot \log n_c \right), \tag{2.1}$$

where n_M is the number of modules in the model, n_c is the total number of components, and M_i is the number of components in the i-th module. The logarithm base in this chapter is 2, and hence the description length is of units of bits. For the example in Figure 2.6, $n_M = 2$, $n_c = 7$, $M_1 = 3$, and $M_2 = 5$. The table in the figure reads as follows: "module 1 has 3 components: A, F, and E; module 2 has 5 components: E, D, B, C, and G." If n_M and n_c are known, it is not difficult to see that the above model description is uniquely decodable. When n_c is given, and assuming $n_M \leq n_c$, then $\log n_c$ bits are needed to describe n_M. The numbers of bits needed to describe n_M and n_c are fixed for all models, and hence are omitted without loss of accuracy. If bus detection is incorporated, the bus is treated exactly as another module. Therefore, its description length is also captured in Equation 2.1.

2.5.3 Mismatched data description

Based on the model, another DSM, $DSM' = [d'_{ij}]$, is constructed where d'_{ij} is 1 *if and only if* (1) some module contains both components i and j simultaneously, or (2) the bus contains either component i or j. Comparing DSM' with the given DSM, for every mismatched



Length	$\log_2 n_c$	$3\log_2 n_c$	$\log_2 n_c$	$5\log_2 n_c$
Description	4	A, F, E	5	E, D, B, C, G

Figure 2.6: Model description and mismatched data. The above is a clustering arrangement of a DSM. The below is the associated model description. The shadowed cells represent mismatches. There are 10 mismatches in this model.

entry, where $d'_{ij} \neq d_{ij}$, a description needs to indicate where the mismatch occurs and whether the mismatch is zero-to-one or one-to-zero. Define the following two mismatch sets: $S_1 = \{(i,j)|d_{ij} = 0, d'_{ij} = 1\}$ and $S_2 = \{(i,j)|d'_{ij} = 1, d'_{ij} = 0\}$. Mismatches that contribute to S_1 are called *type-I mismatch*, and those contribute to S_2 are called *type-II mismatch*. The mismatched data description length is given by:

$$\Sigma_{(i,j)\in S_1} (\log n_c + \log n_c + 1) + \Sigma_{(i,j)\in S_2} (\log n_c + \log n_c + 1)\,. \tag{2.2}$$

The first $\log n_c$ inside the bracket indicates the row number i, the second one indicates the column number j, and the additional one bit indicates the type of the mismatch. The two summations are the description lengths needed for all type-I mismatches and type-II mismatches, respectively. For example, (3, 6, 0) denotes a type-I mismatch in row 3 and column 6; (4, 5, 1) denotes a type-II mismatch in row 4 and column 5.

2.5.4 Putting it all together

The MDL-based clustering metric is given by the summation of the model description length given in Chapter 2.5.2 and the mismatched data description given in Chapter 2.5.3. With

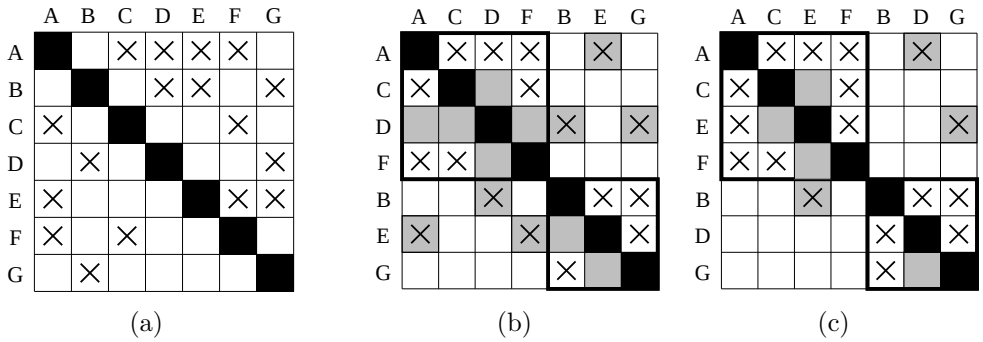


Figure 2.7: An example showing how the metric chooses between different models. Figure 2.7(a) is the original DSM before clustering. Figures 2.7(b) and 2.7(c) are two possible models describing the given DSM. Shadowed cells represent mismatches. Figure 2.7(b) has 4 type-I mismatches and Figure 2.7(c) has 4 type-II mismatches. The MDL-based metric favors the model in Figure 2.7(c) since it is a simpler model with a shorter model description).

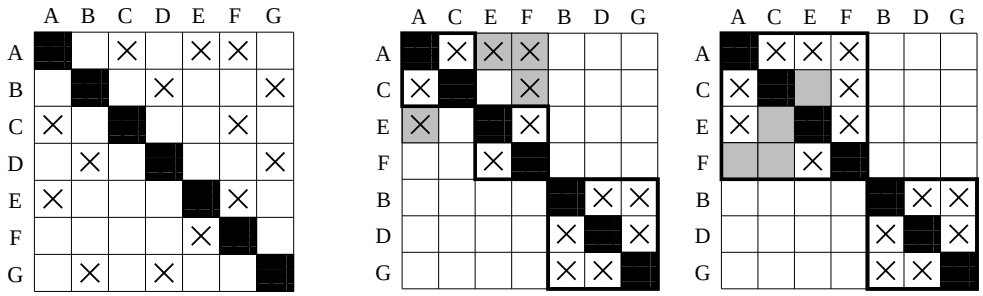
arithmetic manipulation, the metric can be expressed as:

$$f_{DSM}(M) = (n_M \log n_c + \log n_c \Sigma_{i=1}^{n_M} M_i) + (|S_1| + |S_2|) \cdot (2 \log n_c + 1). \tag{2.3}$$

With the above metric, the DSM clustering problem is converted to an optimization problem: Given a DSM, the objective is to find a DSM clustering arrangement, M, to minimize the metric, f_{DSM}. In other words, f_{DSM} is the length needed to describe the given data set DSM by using the model M.

Figure 2.7 illustrates an example on how the metric chooses between different models. Figure 2.7(a) is the original DSM before clustering. Figures 2.7(b) and 2.7(c) are two possible models describing the given DSM. Figure 2.7(b) has 4 type-I mismatches and Figure 2.7(c) has 4 type-II mismatches. The MDL-based metric favors the model in Figure 2.7(c) since it is a simpler model with a shorter model description.

Figure 2.8 shows that MDL prefers fewer mismatches if the models are the same. Figure 2.8(a) shows the original DSM before clustering. The clustering arrangement in Figure 2.8(b) has 3 type-I mismatches and 4 type-II mismatches. Likewise, there are 9 type-I mismatches and 7 type-II mismatches in Figure 2.8(c). Given the model complexities are



(a) (b) (c)

Figure 2.8: If the models are the same, MDL prefers fewer mismatches. Figure 2.8(a) is the original DSM before clustering. Figures 2.8(b) and 2.8(c) are two possible clustering arrangements. The clustering arrangement in Figure 2.8(b) has 3 type-I mismatches (shadowed cells outsider modules) and 4 type-II mismatches (shadowed cells inside modules). Likewise, there are 6 type-I mismatches and 7 type-II mismatches in Figure 2.8(c). Therefore, $\{ACEF\}\{BDG\}$ is a better clustering arrangement than $\{ACDF\}\{BEG\}$ according to MDL.

the same, $\{ACEF\}\{BDG\}$ is a better clustering arrangement than $\{ACDF\}\{BEG\}$ for the DSM, according to MDL.

2.6 Search Strategy: A Simple GA

This section applies the proposed MDL-based clustering metric to a simple genetic algorithm and tests the clustering algorithm on several manually designed examples.

The chromosome is a binary string of $(M_{max}n_c)$ bits, where M_{max} is a predefined maximal number of modules and n_c is the number of components. The $(x + yn_c)$-th bit indicates whether or not the x-th component belongs to the y-th module. The last module represents the bus. Note that the chromosome encoding allows one component to belong to multiple modules, as desired. At the end of the GA run, the best chromosome tells which components belong to which modules or the bus, and it should correspond to a very short description length. In other words, the GA tries to find a model M that minimizes Equation 2.3 for a given DSM.

The simple GA adopts the $(\lambda + \mu)$ selection, uniform crossover, and simple bit-wise

mutation. Note that elitism is embedded in the $(\lambda + \mu)$ selection. The procedure of the specialized GA is outlined as follows.

1. **Create an initial population.** The alleles of the chromosomes are randomly initialized to 0 or 1. The initial population contains λ chromosomes.

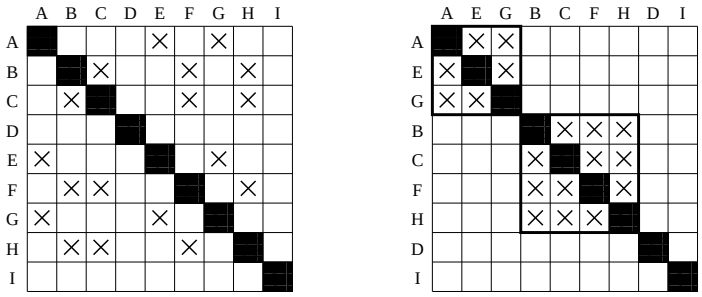
2. **Recombine chromosomes to produce new offspring.** Here, uniform crossover is adopted. The uniform crossover operator randomly switches each gene of the two parent chromosomes with a probability of 0.5 to produce two new offspring chromosomes. If an offspring chromosome takes the best parts from each of its parents, the result will likely be a better solution. The two parents are randomly chosen without replacement from the initial λ chromosomes, and the reproduction is continued until μ offspring chromosomes are produced. Note that no selection is performed here yet.

3. **Mutate the genes of the offspring chromosomes.** For a binary chromosome, the bit-wise mutation inverts the values of genes with a mutation probability p_m.

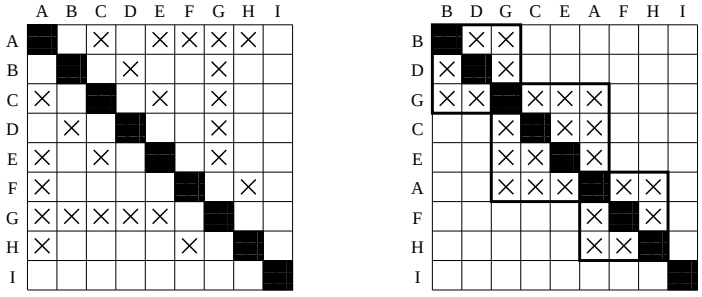
4. **Evaluate chromosomes.** Each chromosome is evaluated according to the MDL-based clustering metric to determine the solution quality. Note that only μ chromosomes need to be evaluated except for the first generation.

5. **Perform $(\lambda + \mu)$ selection.** The $(\lambda + \mu)$ selection operator selects λ best chromosomes from all $(\lambda + \mu)$ chromosomes and passes them to the next generation. Note that elitism is embedded in $(\lambda + \mu)$ selection.

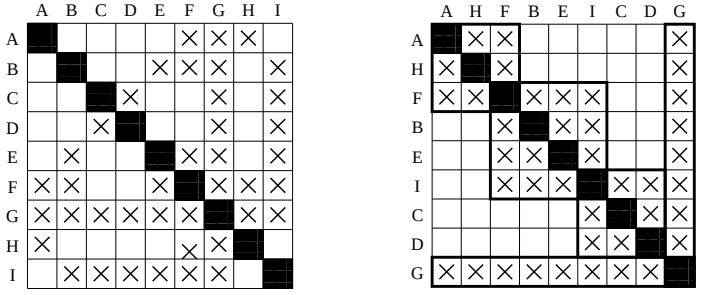
6. **Repeat steps 2 through 5 until termination.** In this chapter, the GA terminates when no improvement occurs within a certain number of generations.



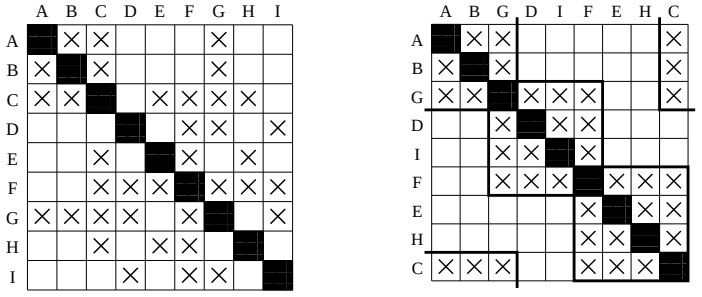
(a) Two non-overlapping modules.



(b) Three overlapping modules.



(c) Three overlapping modules with a bus.



(d) Three overlapping modules; one is broken.

Figure 2.9: Illustrative examples of DSM clustering. The left column is the manually designed DSMs given as inputs to the GA. The right column shows how the GA clusters them.

2.6.1 Illustrative examples

The MDL-based metric is tested on several manually designed problems to illustrate the abilities of the proposed DSM clustering algorithm. In the following experiments, the number of components is 9, the maximal number of modules is set to 4, and hence the chromosome length is $9 \times 4 = 36$. The crossover probability p_c is 1, mutation probability p_m is $\frac{1}{36}$, and a (50+50) selection is adopted. The GA terminates if no improvement happens in five consecutive generations. The weights in the MDL-based clustering metric are set equally at $\frac{1}{3}$. Figure 2.9 shows the GA results of the examples. Those DSMs are designed with known optimal clustering arrangements. They are then randomly reordered and handed to the GA as inputs to test if the GA is able to rediscover the modules back. Figure 2.9(a) shows a simple case with two non-overlapping modules. Figure 2.9(b) demonstrates the ability to identify overlapping modules. In Figure 2.9(c), a bus is introduced, and the GA is able to identify it. Note that the DSM in Figure 2.9(d) resembles the DSM in Figure 2.9(c); however, the results are totally different. The GA recognizes the DSM in Figure 2.9(d) as three overlapping modules instead of a bus. Figure 2.9(d) also demonstrates the ability to identify modules that are broken in the 2D representation. In summary, these results show that the GA with the MDL-based clustering metric is able to correctly cluster DSMs with overlapping modules, a bus, or three dimensional structures.

2.7 From Binary DSM to Weighted DSM

Most real-world DSMs are real-valued or contain information of different levels of interaction strength. Therefore, it is necessary to extend the algorithm to perform clustering on weighted DSMs. By normalizing a weighted DSM, the value of each entry can be considered as the probability of communication. This idea is consistent with binary DSMs. In a binary DSM, $d_{ij} = 1$ can be thought as having component i communicates with component j with probability 1. The same interpretation is also valid for $d_{ij} = 0$. Based on the interpretation,

modification to the MDL-based clustering metric can be made as follows. First, entry d_{ij} in the DSM is normalized to $p_{ij} = \frac{d_{ij}-d_{min}}{d_{max}-d_{min}}$, where $d_{max} = \max_{i,j} d_{ij}$ and $d_{min} = \min_{i,j} d_{ij}$. The formula for the description length of model complexity remains the same. The mismatch sets for type I and type II are modified as $S_1 = \Sigma_{\{(i,j)|d'_{ij}=1\}} (1 - p_{ij})$ and $S_2 = \Sigma_{\{(i,j)|d'_{ij}=0\}} p_{ij}$, respectively. The modification comes from entry d_{ij} having a probability of $(1 - p_{ij})$ to be a type-I mismatch if it is inside a module and a probability of p_{ij} to be a type-II mismatch if it is outside any modules.

2.8 Tuning the Clustering Metric to Mimic Human Preference

This section demonstrates how to tune the MDL-based metric to mimic human experts' preference. Empirical findings show that human experts tend to give clustering arrangements with sparse modules to eliminate the interactions outside all the modules (Yu, Yassine, & Goldberg, 2005). The reason behind such preference may lie on some factors not captured in DSMs. To mimic this behavior, the MDL-based metric can be weighted accordingly.

Instead of using summation, a weighted summation can be used for the clustering metric, which yields

$$\begin{aligned} f_{DSM}(M) &= w_1 \cdot (n_M \log n_c + \log n_c \Sigma_{i=1}^{n_M} M_i) \\ &+ w_2 \cdot |S_1|(2 \log n_c + 1) + w_2 \cdot |S_2|(2 \log n_c + 1), \end{aligned} \tag{2.4}$$

where w_1, w_2 and w_3 are weights between 0 and 1 with $w_1 + w_2 + w_3 = 1$. Without any prior knowledge or assumption, w_i can be set at $\frac{1}{3}$ as in previous sections. This section utilizes *Widrow-Hoff iteration* (Widrow & Hoff, 1960) to tune the weights in Equation 2.4, so that the GA results would have a similar ratio of the description lengths preferred by human experts.

For simplicity, rewrite Equation 2.4 as:

$$f_{DSM}(M) = w_1 f_1 + w_2 f_2 + w_3 f_3. \tag{2.5}$$

The objective of adjusting the weights is to find w_i such that at the end of the GA run, $\frac{f_1}{f_1+f_2+f_3} : \frac{f_2}{f_1+f_2+f_3} : \frac{f_3}{f_1+f_2+f_3} \simeq r_1 : r_2 : r_3$, where r_i reflects the human experts' preference of clustering arrangements (Chapter 2.9). Since the summations of the weights and ratio are both ones, this is a nonlinear system with two degrees of freedom. Define two nonlinear functions as follows.

$$\begin{cases} R_1(w_1, w_2) = \frac{f_1(w_1,w_2)}{f_1(w_1,w_2)+f_2(w_1,w_2)+f_3(w_1,w_2)} - r_1. \\ R_2(w_1, w_2) = \frac{f_2(w_1,w_2)}{f_1(w_1,w_2)+f_2(w_1,w_2)+f_3(w_1,w_2)} - r_2. \end{cases} \tag{2.6}$$

Now, the objective of adjusting weights is to find a pair (w_1, w_2) such that $(R_1, R_2) \simeq (0, 0)$. There have been many numeric methods trying to conquer nonlinear systems such as Newton-Raphson method (Press, Teukolsky, Vetterling, & Flannery, 1992), Broyden method (Broyden, 1965), and the steepest descent method (Press et al., 1992). However, none of the above is suitable for the problem here. Newton-Raphson method requires the information of first derivatives, not known in this case; Broyden method utilizes difference equations to approximate the gradient information on-the-fly, which in this case, increases the noise produced by the GA; the steepest descent method converges slowly and requires much more GA runs. Therefore, based on the following assumption, the gradient information is derived to guide the Widrow-Hoff iteration.

Define $c_1 = w_1 f_1$ and $c_2 = w_2 f_2$. Assume that for a small change of w_i, c_i does not vary much. The assumption is reasonable because for a larger w_i, the GA will try to find a solution with a smaller f_i to minimize $w_i f_i$. In other words, the following Jacobian matrix

is assumed to be zero.

$$\begin{bmatrix} \frac{\partial c_1}{\partial w_1} & \frac{\partial c_2}{\partial w_1} \\ \frac{\partial c_2}{\partial w_1} & \frac{\partial c_2}{\partial w_2} \end{bmatrix} = \mathcal{O}, \tag{2.7}$$

where $\mathcal{O}$ is the zero matrix. Based on this assumption, R_1 and R_2 are explicit functions of w_1 and w_2.

$$\begin{cases} R_1 = \frac{f_1}{f_1+f_2+f_3} - r_1 = \frac{c_1}{w_1\left(\frac{c_1}{w_1}+\frac{c_2}{w_2}+\frac{c_3}{1-w_1-w_2}\right)} - r_1. \\ R_2 = \frac{f_2}{f_1+f_2+f_3} - r_1 = \frac{c_2}{w_2\left(\frac{c_1}{w_1}+\frac{c_2}{w_2}+\frac{c_3}{1-w_1-w_2}\right)} - r_2. \end{cases} \tag{2.8}$$

The following Jacobian matrix can be calculated.

$$J = \begin{bmatrix} \frac{\partial R_1}{\partial w_1} & \frac{\partial R_2}{\partial w_1} \\ \frac{\partial R_1}{\partial w_2} & \frac{\partial R_2}{\partial w_2} \end{bmatrix} = \begin{bmatrix} \frac{-c_1\triangle - c_1 w_1 \triangle_1}{w_1^2 \triangle^2} & \frac{-c_2\triangle_1}{w_2\triangle^2} \\ \\ \frac{-c_1\triangle_2}{w_1\triangle^2} & \frac{-c_2\triangle - c_2 w_2 \triangle_2}{w_2^2\triangle^2} \end{bmatrix}, \tag{2.9}$$

where $\triangle = \frac{c_1}{w_1}+\frac{c_2}{w_2}+\frac{c_3}{w_3}$, $\triangle_1 = \frac{-c_1}{w_1^2}+\frac{c_3}{w_3^2}$, $\triangle_2 = \frac{-c_2}{w_2^2}+\frac{c_3}{w_3^2}$, and $c_3 = w_3 f_3$. Since $[\partial w_1 \quad \partial w_2]\cdot J = [\partial R_1 \quad \partial R_2]$, the Widrow-Hoff iteration is given as follows.

$$[w_1 \quad w_2]_{(t+1)} = [w_1 \quad w_2]_{(t)} - \eta\,[R_1(w_1,w_2) \quad R_2(w_1,w_2)]_{(t)} \cdot J_{(t)}^{-1}, \tag{2.10}$$

where t is the iteration index, and η is the learning rate. A higher learning rate enables a faster learning; however, it may cause the iteration to diverge. The iteration given in Equation 2.10 can be used as follows to find the weights that yield $\frac{f_1}{f_1+f_2+f_3} : \frac{f_2}{f_1+f_2+f_3} : \frac{f_3}{f_1+f_2+f_3} \simeq r_1 : r_2 : r_3$. Given a starting point $[w_1 \quad w_2]_{(0)}$ and after the GA gives f_1, f_2, and f_3, R_1, R_2, and the Jacobian matrix are calculated according to Equations 2.8 and 2.9. Afterward, Equation 2.10 gives the weights for the next GA trial. If the Widrow-Hoff iteration converges, having more iterations should produce a ratio closer to the objective deducted from human experts' preference.

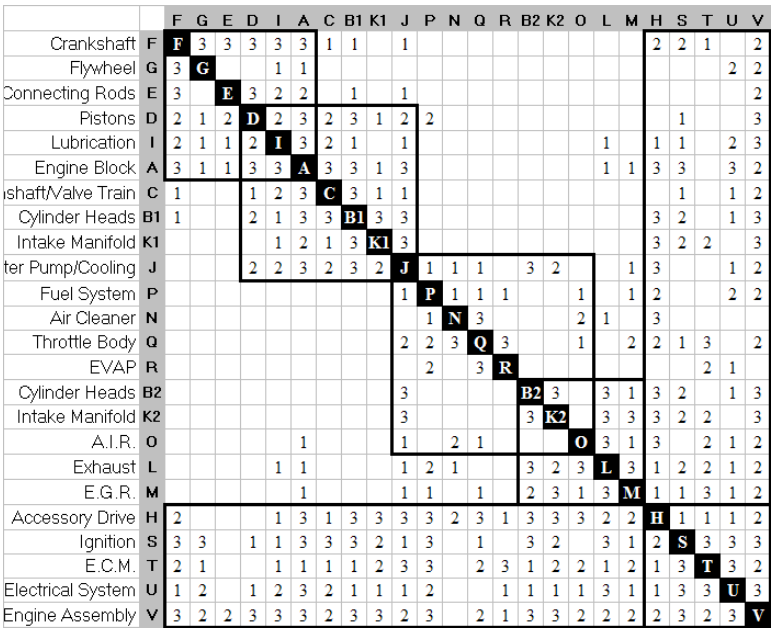


Figure 2.10: Clustering arrangement by human expert for the GMPT DSM system teams.

2.9 Real-world Case Study

In this section, the proposed DSM clustering algorithm is tested on a real-world DSM—a DSM for the GM power train development teams (GMPT) (McCord & Eppinger, 1993; Eppinger, 2001). It is a DSM for the communication patterns between 22 engine product development teams at GM constructed by circulating a survey about communication frequency among teams. These 22 teams are arranged as 4 modules and a bus by manual clustering of the DSM shown in Figure 2.10[1] (McCord & Eppinger, 1993; Eppinger, 2001). Note that components B and K are repeated twice, shown as $B1$, $B2$, $K1$ and $K2$, in the clustering arrangement to enhance the visualization.

2.9.1 Automated clustering using the proposed MDL-GA

Having the maximal number of modules set to half of the number of components, the chromosome length for clustering the GMPT DSM is $11 \times 22 = 132$. The crossover proba-

[1]The monthly, weekly, and daily interactions (McCord & Eppinger, 1993) are transferred to 1, 2, and 3, respectively.

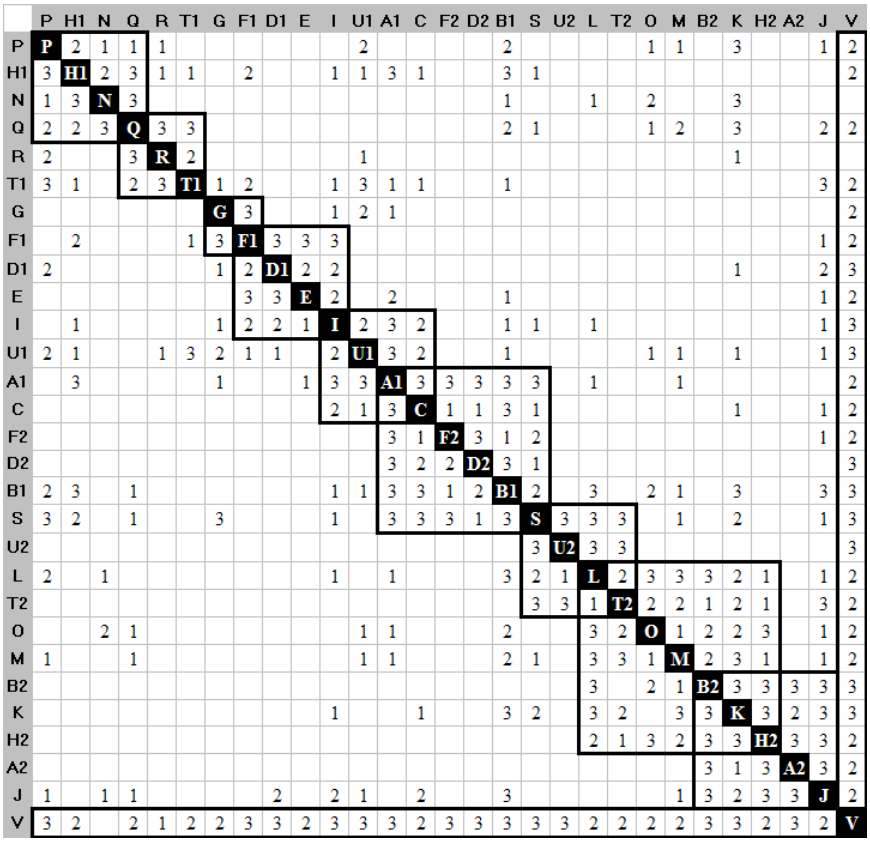


Figure 2.11: Clustering arrangement for the GMPT DSM by the MDL-GA with uniform weights.

bility is set to 0.9, and the mutation probability is set to $\frac{1}{132}$. The GA terminates if no improvement occurs in 50 consecutive generations. A number of experiments show that (5000+5000) selection produces satisfactory results. With the straightforward weight setting, $w_1 = w_2 = w_3 = \frac{1}{3}$, the MDL-GA results in the clustering arrangement in Figure 2.11.

The MDL-GA with uniform weights results in a more complicated model with many denser modules compared to the clustering arrangement given by the human experts. The ratio between the description lengths of the model, type-I mismatch, and type-II mismatch given by the manual clustering arrangement (Figure 2.10) is found to be 0.0933:0.8529:0.0538 [2], while the ratio given by the MDL-GA is 0.1987:0.3683:0.4330.

[2]In the work of Yu, Yassine, and Goldberg (2005), they investigated three manual clustering arrangements of real-world DSMs and found the average ratio of the description lengths of the model, type-I mismatch, and type-II mismatch to be 0.0784:0.8116:0.1102.

	Weights			Description-length ratio		
Objective				0.0933	0.8529	0.0538
Straightforward Setting	0.3333	0.3333	0.3333	0.1987	0.3683	0.4330
Weight Adjusting	0.3512	0.1628	0.4860	0.0904	0.8589	0.0507

Table 2.1: The weights and ratio given by the MDL-GA with uniform weights and with weight adjusting. Compared with the straightforward weight setting, the Widrow-Hoff iteration is able to produce the weights that yields a description-length ratio close to the objective.

	G	T1	F	A	D	E	I	C	J	K	N	O	L	M	P	Q	T2	R	B	H	S	U	V
G	G		3	1			1															2	2
T1	1	T1	2	1			1	1											1	1	3	3	2
F	3	1	F	3	3	3	3	1	1										1	2	2		2
A	1		3	A	3	1	3	3	3	1			1	1					3	3	3	3	2
D	1		2	3	D	2	2	2	2	1					2				3		1		3
E			3	2	3	E	2		1										1				2
I	1		2	3	2	1	I	2	1				1						1	1	1	2	3
C			1	3	1		2	C	1	1									3		1	1	2
J				3	2		2	2	J	2	1			1	1	1			3	3		1	2
K				2			1	1	3	K			3	3			2		3	3	2		3
N										3	N	2	1		1	3			1	3			
O				1					1	2	2	O	3	1		1	2		2	3		1	2
L				1			1		1	2	1	3	L	3	2		2		3	1	2	1	2
M				1					1	3		1	3	M	1	1	3		2	1	1	1	2
P									1	3	1	1		1	P	1		1	2	2		2	2
Q									2	3	3	1		2	2	Q	3	3	2	2	1		2
T2									3	2		2	1	2	3	2	T2	3	1	1	3	3	2
R										1					2	3	2	R				1	
B			1	3	2		1	3	3	3		2	3	1	2	1			B	3	2	1	3
H		1	2	3			1	1	3	3	2	3	2	2	3	3	1	1	3	H	1	1	2
S	3	3	3	3	1		1	3	1	2			3	1	3	1	3		3	2	S	3	3
U	2	3	1	3	1		2	2	1	1		1	3	1	2		3	1	1	1	3	U	3
V	2	2	3	3	3	2	3	2	2	3		2	2	2	3	2	2	1	3	2	3	3	V

Figure 2.12: Clustering arrangement for the GMPT DSM by the proposed GA. The weights are tuned by the Widrow-Hoff iteration.

Instead of using uniform weights, the weight-adjusting technique described in Chapter 2.8 can be adopted to produce weights that yield a similar description-length ratio to the manual clustering arrangement. The number of the Widrow-Hoff iterations is limited to 50. After 50 iterations, the best run is chosen according to the minimal sum of squared errors, $(R_1^2 + R_2^2 + R_3^2)$. The objective ratio and the ratios given by the MDL-GA with uniform weights and with weight adjusting are shown in Table 2.1. The MDL-GA with weight adjusting is able to produce a clustering arrangement with a description-length ratio close to the objective.

	Model	Type-I mismatch	Type-II mismatch
Manual	160.54	1467.99	92.58
MDL-GA with uniform weights	222.97	413.29	486.02
MDL-GA with weight adjusting	147.16	1398.56	82.66

Table 2.2: The description lengths of the DSM clustering arrangements done by human experts versus by the GA.

	Mean of module densities	Variance of module densities
Manual	0.5747	0.0268
MDL-GA with uniform weights	0.8174	0.0093
MDL-GA with weight adjusting	0.5849	0.0065

Table 2.3: The means and variances of the normalized module densities of the manual clustering arrangement and the GA results.

Finally, the clustering arrangement given by the MDL-GA with weight adjusting is shown in Figure 2.12.

2.9.2 Discussion of results

The MDL-GA with uniform weights gives a more complicated model with denser modules and more interactions left outside the modules than the manual clustering, while the MDL-GA with weight adjusting gives a similar preference as the manual clustering. Table 2.2 shows the description lengths in each category for the GMPT DSM clustering arrangement performed by human experts and the proposed MDL-GA algorithm. As the table indicates, the MDL-GA with weight adjusting outperforms the manual clustering in all three categories. In other words, the proposed method gives a simpler model that better describes the GMPT DSM than the human expert.

The proposed method also gives more consistent results than human experts (Yu, Yassine, & Goldberg, 2005). Table 2.3 shows the normalized densities of the modules and their means and variances for both the manual clustering arrangement and the GA results. The MDL-GA with uniform weights gives an arrangement with denser modules, while the MDL-GA with weight adjusting gives an arrangement with modules of similar densities as that of

the manual arrangement. Both the clustering arrangements given by the MDL-GA contain modules with more uniform densities than the manual one.

The results can be interpreted from two angles. If the MDL-based metric is a more appropriate criterion for the clustering problem, the GA provides better solutions than human. On the other hand, if human clustering is more appropriate due to considering several subtle constraints not observed by the GA, then the problem relies on how to tune the MDL-based metric to mimic human experts' preference. Tuning the weights according to the method described in Chapter 2.8 accomplishes such task as an initial attempt. The proposed method provides a consistent, systematic, and automatic way to cluster DSMs, and the clustering results can be either used directly, or used as an initial clustering arrangement for human experts to tune.

2.10 Summary and Conclusions

This chapter starts by reviewing the matrix clustering literature related to product architecture and modularity. Based on the MDL concept, it then proposes a clustering metric. The MDL-based metric is combined with a simple GA to cluster weighted DSMs. Several illustrative examples show that the proposed MDL-GA is capable of clustering DSMs with overlapping modules, a bus, and modules with a three dimensional structure. Finally, results from applying the MDL-GA to a real-world problem show the promise of automated clustering using GAs.

The DSM is a powerful tool for representing and visualizing product architectures. This representation allows for the analysis and development of modular products by clustering the DSM. Existing clustering algorithms are scarce and inefficient especially when confronted with complex product architectures as in the reported case studies. The MDL-GA clustering algorithm presented in this chapter is capable of identifying complex clustering arrangements and overcoming many of the difficulties observed in existing clustering tools. Using MDL

as a principle to cluster DSMs is more systematic than manual clustering, and the results are promising. Nevertheless, the proposed algorithm can be further refined. These efforts include an extension to multi-objective clustering, where the values of entries in the DSM represent different types of dependencies between two components (Schloegel, Karypis, & Kumar, 1999). Along similar lines, a more explicit representation of domain-specific expert knowledge may allow for better tuning of the weights of the model description, and more experiments might be needed to capture the real preference of human experts (Nascimento & Eades, 2001). Furthermore, the proposed method is capable of identifying buses and overlapping modules, but other predominant architectural features may also need to be identified and incorporated into the MDL clustering metric. One such example is the concept of a mini-bus or a floating bus discussed by Sharman and Yassine (2004). Finally, the proposed MDL-based clustering metric could be used in conjunction with other search algorithms for the different needs of clustering speed and quality.

Comparison using known problem instances should be done to existing and the proposed clustering methods in order to verify that the proposed clustering metric and the associated GA are superior in extracting "optimal" modular arrangements from DSMs. However, the proposed approach is unique in many ways, and hence simple and direct comparisons are unavailable. The unique features of the proposed clustering method lie in the following features:

1. Accounting for bus modules.
2. Detecting overlapping modules.
3. Having a unique information theoretic clustering metric.
4. Tuning the clustering metric to mimic human experts' preference.
5. Generating reusable weighting parameters for similar products.

In closing, the algorithm presented in this chapter might introduce a series of rigorous mathematical clustering algorithms. This should lead to improved products, processes, and organizational designs. Interestingly, the MDL-DSM combination also leads the way to investigate the dual problem of using DSM clustering to design more effective genetic algorithms (Yu, Goldberg, Yassine, & Chen, 2003). The rest of this thesis will discuss this topic in greater details.

Chapter 3

Finding Extrema for Problems with Modularity: DSMGA

This chapter describes how the proposed dependency structure matrix genetic algorithm (DSMGA) solves problems with non-overlapping modularity. DSMGA utilizes the dependency structure matrix (DSM) clustering techniques from the previous chapter to extract building blocks (BBs) information and uses the information to accomplish BB-wise crossover. Here a BB is a group of highly interacting genes, and is essentially a module. Three cases: tight, loose, and random modularity, are tested using DSMGA and a simple GA. Empirical results show that DSMGA is able to correctly identify BBs and outperforms a simple GA by using the extracted BB information.

3.1 Interaction Detection

It has been shown that if the GA is not capable of learning the interaction topology of the problem, the population size required for finding the global optima scales exponentially with the problem size (Thierens & Goldberg, 1993). If the GA is able to learn the problem structure and decompose the problem accordingly, the population size required is $O(m2^k)$, where m is the number of BBs and k is the order of one BB. In other words, a GA with proper problem decomposition only needs a polynomial number of function evaluations to the problem size for boundedly difficult problems, where k is bounded by a constant.

The key to reduce the run duration of GAs from exponential to polynomial is interaction detection. Many such techniques have since been developed and can be categorized as follows.

Perturbation. GAs in this category detect interactions among genes by perturbing alleles and monitoring the change done by such perturbation. Such GAs usually consist of two phases. The first phase perturbs alleles and detects interactions, and the second phase combines the promising building blocks. Typical examples are mGA (Goldberg, Korb, & Deb, 1989), fmGA (Goldberg, Deb, Kargupta, & Harik, 1993), gemGA (Kargupta, 1996), LINC and LIMD (Munetomo & Goldberg, 1999).

Interaction adaptation. GAs in this category utilize a unique metric–fitness to detect interactions and solve the problem at the same time. The interaction arrangement is embedded in the encoding. A chromosome with highly interacting genes encoded closer has a higher survival rate under recombination. Typical examples are LEGO (Smith & Fogarty, 1996) and LLGA (Harik, 1997).

Model building. The typical framework of the GAs in this category has a model-building phase after selection. A typical model-building GA consists of the following steps: (1) randomly initializing a new population, (2) selecting a set of promising solutions, (3) model building based on those promising solutions, (4) generating the next generation by utilizing the model, and (5) repeating steps 2 to 5 until the termination condition is met. Examples are cGA (Harik, Lobo, & Goldberg, 1998), ecGA (Harik, 1999), BOA (Pelikan, Goldberg, & Cantú-Paz, 1999), and hBOA (Pelikan & Goldberg, 2001). More examples can be found in Larrañaga and Lozano (2002) and Goldberg (2002).

DSMGA developed in this thesis belongs to the model-building GA category. Among GAs in this category, DSMGA resembles ecGA and hBOA the most while differs in the following aspects. ecGA uses an information theoretical metric to detect the interactions among genes. The interaction model used by ecGA is explicit, but does not consider overlap and hierarchy. hBOA utilizes a Bayesian network to express the interactions among genes and implicitly handles problems with modularity, hierarchy, and overlap. Similar to ecGA, the interaction model in the DSMGA design is also explicit, but the DSMGA design handles

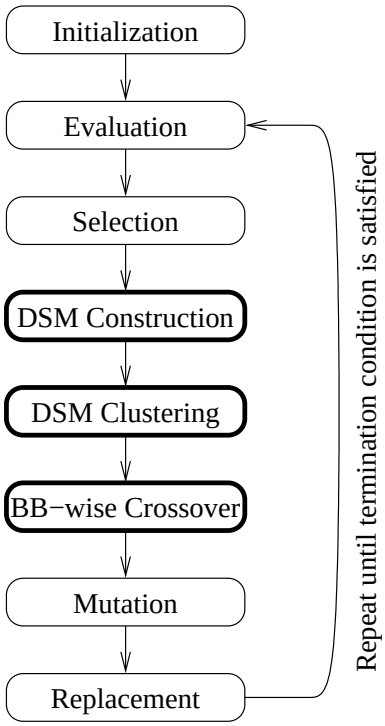


Figure 3.1: The framework of DSMGA. The major difference between DSMGA and a simple GA is that DSMGA utilizes DSM clustering techniques to obtain BB information and then uses it to achieve BB-wise crossover.

problems with modularity, hierarchy (Chapter 5), and overlap (Chapter 7). hBOA also solves problems with all three types of interactions, but the interaction model in DSMGA is more comprehensible so it is easier for researchers to gain knowledge about the problem.

3.2 Framework of DSMGA

There are two optimization problems in DSMGA. One is the given optimization problem, and the other is the DSM clustering problem. The key idea of DSMGA is to use an auxiliary search algorithm to extract the BB information by using the DSM clustering technique, and then solve the given optimization problem more effectively and efficiently by using the extracted BB information. The main task in DSMGA can be decomposed into three subtasks: (1) DSM construction from the current population, (2) DSM clustering, and (3) BB-wise crossover. The framework of DSMGA is shown in Figure 3.1.

The DSM clustering has been detailed in the previous chapter. The only difference in DSMGA is that detection of the bus module is not allowed and the weights in the MDL-based metric are fixed to the straightforward setting where $w_i = 1/3$. Also, based on the research by Yu and Goldberg (2004), it is beneficial to adopt a hill-climber for the DSM clustering to save computational time while not degrading much model quality.

BB-wise crossover (Harik, 1999) is performed after DSM clustering obtains BB information. The BB-wise crossover is similar to a regular allele-wise crossover when there is no overlapping BBs. The only difference is that the exchange of information is performed at the BB level. If the BB information is perfect, no BB disruption occurs. DSMGA adopts BB-wise uniform crossover since it provides a higher mixing rate than one-point crossover or two-point crossover.

The only remaining task is DSM construction, and it will be discussed in the next section.

3.3 DSM Construction

Interaction-detection mechanism is essential for constructing DSMs. Many different metrics have been used for interaction detection in GAs. This section first compares and contrasts three commonly used metrics—nonlinearity, simultaneity, and entropy. Among them, the entropy metric is chosen to construct DSMs in DSMGA. By investigating the distribution of sampled entropy, the constructed DSM is converted into a binary DSM to alleviate computational burden.

3.3.1 Interaction-detection Metrics

Of all the interaction-detection metrics, the following three are commonly used.

Nonlinearity is adopted in LINC and LIMD (Munetomo & Goldberg, 1999). When focusing on only two genes x_i and x_j, as in the case of DSMGA, the detection metric can

be written as

$$f_{x_i=0,x_j=0} + f_{x_i=1,x_j=1} - f_{x_i=1,x_j=0} - f_{x_i=0,x_j=1}. \tag{3.1}$$

The basic idea is that if x_i and x_j do not interact with each other, the fitness difference from $x_i = 0$ to $x_i = 1$ should not be affected by the value of x_j.

Simultaneity is adopted in the work of Aporntewan and Chongstitvatana (2003). The idea is to consider the BB disruptions when 00 crosses with 11 or 10 crosses with 01. The detection metric can be expressed as

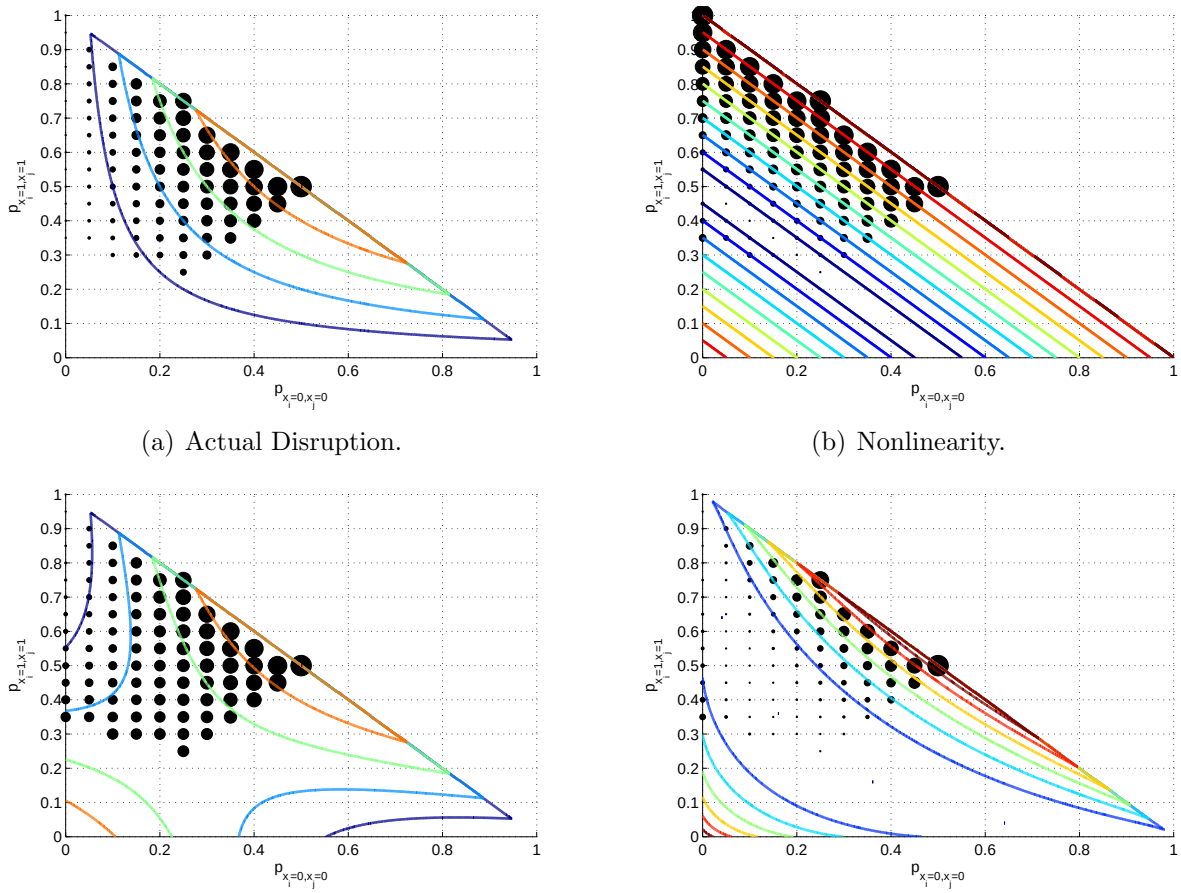
$$p_{x_i=0,x_j=0}p_{x_i=1,x_j=1} + p_{x_i=0,x_j=1}p_{x_i=1,x_j=0}, \tag{3.2}$$

where p is the proportion of the event in the current population.

Entropy-based methods are most commonly used for interaction detection in GAs. ecGA (Harik, 1999), BMDA (Pelikan & Mühlenbein, 1999), and BOA (Pelikan, Goldberg, & Cantú-Paz, 1999) are some typical examples. For the case of two genes, the loss in entropy is mutual information. The idea is to measure the certainty of x_j given the information of x_i. The metric can be written as

$$\Sigma_{x_i,x_j} p_{x_i,x_j} \log \frac{p_{x_i,x_j}}{p_{x_i} p_{x_j}}. \tag{3.3}$$

The following scenario compares these detection metrics by considering two genes. A proportionate selection is assumed so that the proportion of a schema is proportional to its fitness value ($p_{x_i=a_i,x_j=a_j} \propto f_{x_i=0,x_j=0}$). Suppose that the correct BB is 11. It will be disrupted only when crossed with 00. Since 11 is the correct BB, $p_{x_i=1,x_j=1}$ can be assumed to be greater than the other three. For simplicity, also assume $p_{x_i=0,x_j=1} = p_{x_i=1,x_j=0}$. Figure 3.2 gives the comparison of these three detection metrics for different possible values of $p_{x_i=1,x_j=1}$ and $p_{x_i=0,x_j=0}$.



(a) Actual Disruption.

(b) Nonlinearity.

(c) Simultaneity.

(d) Mutual Information.

Figure 3.2: The comparison of different interaction-detection metrics. The Y-axis is the proportion of the correct BB; the X-axis is the proportion of its most competing BB. The sizes of dots indicate the signal strengths detected. Signals on the same contour have the same strength. Figure 3.2(a) is the ideal case. Among the three metrics, nonlinearity is the most different. Both simultaneity and mutual information are closer to the actual disruption.

As the figure indicates, nonlinearity differs the most among the three metrics. Both simultaneity and mutual information are closer to the actual disruption. Nonlinearity emphasizes the disruptions more than needed when the proportion of the correct BB is much greater than the other three schemata. To show how this can happen, define the fitness as a power-law of the OneMax problem, $f(\vec{x}) = (\Sigma_i x_i)^\kappa$, where x_i is binary and κ is an parameter of order. The problem can be solved by a bit-wise hill climber; however, the nonlinearity metric would detect strong interactions between all genes when κ is large. The simultaneity metric slightly overestimates the disruptions when $p_{x_i=1,x_j=1}$ and $p_{x_i=0,x_j=0}$ are both small.

On the contrary, the mutual information metric slightly underestimates the disruptions in the same region. Nevertheless, both metrics detect the disruptions well around the region where $p_{x_i=1,x_j=1} \simeq p_{x_i=0,x_j=0}$, and that is also where the actual disruptions occur often and need to be taken care of.

The first version of DSMGA (Yu, Goldberg, Yassine, & Chen, 2003) uses nonlinearity to detect the interactions. However, the above discussions indicate that simultaneity and mutual information should be better metrics in terms of detecting disruptions. In this thesis, constructing the DSM in DSMGA is based on the mutual information metric.

3.3.2 Interaction-detection threshold

To alleviate computational burden, after the sampled mutual information is calculated, it is transferred into the binary domain where the metric indicates whether or not the pair of genes interact with each other. Once an appropriate threshold is calculated, a pair of genes is considered as interacting with each other if and only if the calculated mutual information is greater than the threshold. The threshold can be decided by investigating the distribution of mutual information.

Mutual information is defined as the Kullback-Leibler distance (Kullback & Leibler, 1951) between the joint distribution and the product distribution:

$$\begin{aligned} \mathbb{I}(X;Y) &= D(p(x,y)||p(x)p(y)) \\ &= \Sigma_x \Sigma_y p(x,y) \log \tfrac{p(x,y)}{p(x)p(y)}, \end{aligned} \tag{3.4}$$

where D is the Kullback-Leibler distance, X and Y are two random variables, and x and y are the outcomes of these two random variables, respectively. Note that if X and Y are independent, $p(x,y) = p(x)p(y)$, and hence $\mathbb{I}(X;Y) = 0$.

According to Kleiter (1999) and Hutter and Zaffalon (2005), if two random variable X and Y are independent, the sampled mutual information $\mathbb{I}_n(X;Y)$ calculated over n samples

has a mean $\mu \simeq \frac{1}{2n}$ and a variance $\sigma^2 \simeq \frac{1}{2n^2}$. Both μ and σ^2 tends to zero when the sample size n is large.

Approximate the distribution of the mutual information by a Gaussian distribution (Hutter & Zaffalon, 2005). Given a threshold θ, the decision error ϵ is given by $1 - \Phi(z)$, where Φ is the cumulative standard Gaussian function and $z = \frac{\theta - \mu}{\sigma}$.

For a problem with m modules, identifying at least $(m-1)$ modules correctly is preferred. Thus the overall decision accuracy needs to be greater than or equal to $\frac{m-1}{m}$. In a DSM, there are approximately $\frac{l^2}{2}$ pairs of genes. Therefore, the following relation holds.

$$(1 - \epsilon)^{\frac{l^2}{2}} \geq \frac{m-1}{m}. \tag{3.5}$$

For a large m and a large l, Inequality 3.5 can be approximated as $\epsilon \leq \frac{2}{ml^2}$.

When the decision variable z is large, the cumulative standard Gaussian function can be approximated using the first term of the Taylor series of the error function, which yields the following inequality.

$$\frac{e^{-\frac{z^2}{2}}}{\sqrt{2\pi}z} \leq \frac{2}{ml^2}. \tag{3.6}$$

Recall that $z = \frac{\theta - \mu}{\sigma}$. Given $l = km$, Inequality 3.6 can be solved for z as

$$z \geq \sqrt{\mathbb{W}(cl^6)}, \tag{3.7}$$

where $\mathbb{W}$ is Lambert's W-function[1] and $c = \frac{1}{8\pi k^2}$. For problems with unknown k, it is conservative to calculate z by assuming $m = l$, and hence $c \simeq 0.04$. Given a population size of n, the threshold that ensures $(m-1)$ modules being identified correctly can be decided as

$$\theta = \frac{1}{2n} + \sqrt{\frac{\mathbb{W}(cl^6)}{2n^2}}. \tag{3.8}$$

The mutual information metric can also be transferred into probabilistic domain instead

[1]Lambert's W-function is defined as the inverse function of $f(W) = We^W$

of binary domain. In this case, the transferred metric indicates the probability that the pair of genes interact with each other. However, it is not necessary given that the variance is approximately $\frac{1}{2n^2}$, which decreases rapidly with population size n increasing. In a regular GA experiment, the population size is usually large enough such that the *grey area* is small, and the information from the binary domain is accurate enough to make the correct decision.

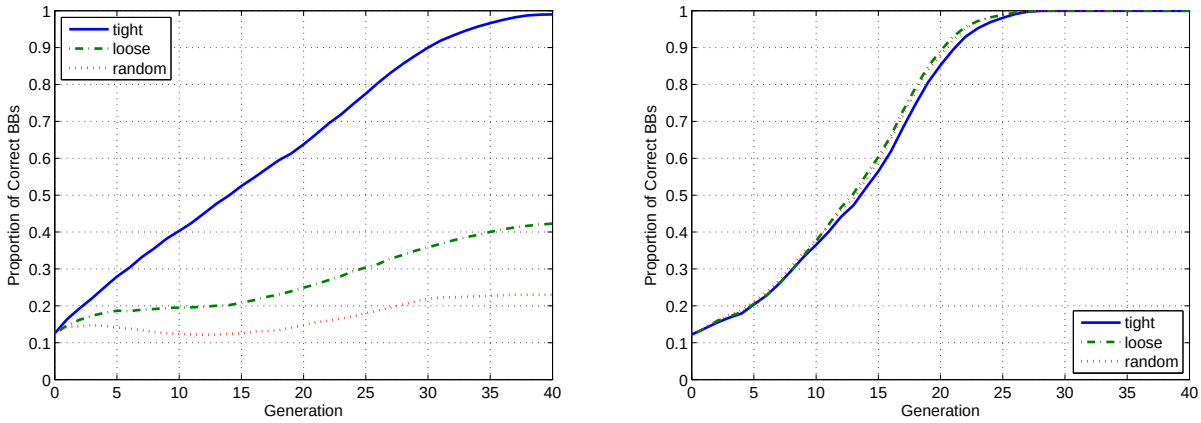
3.4 Modularity Identification Test

The test function is a (m, k)-trap, where $m = 10$ and $k = 3$. The 3-bit trap is given by $f^3_{trap}(u = 0) = 0.9$, $f^3_{trap}(u = 1) = 0.45$, $f^3_{trap}(u = 2) = 0.0$, and $f^3_{trap}(u = 3) = 1.0$, where u is the number of 1's. The key idea of the design behind the trap function is to emphasize on problem decomposition. If the problem is not properly decomposed, any hill-climbing methods tend to give a solution containing all 0's (Goldberg, 1987).

Three cases were tested: tight, loose, and random modularity. Define $U(x)$ as a counting function that counts the number of 1's in x. In the tight modularity test, interacting genes are arranged next to each other, and the fitness function is define as $f^3_{trap}(U(x_1, x_2, x_3)) + f^3_{trap}(U(x_4, x_5, x_6)) + f^3_{trap}(U(x_7, x_8, x_9)) + \cdots$. In the loose modularity test case, interacting genes are arranged away from each other as farther as possible, and the fitness function is defined as $f^3_{trap}(U(x_1, x_{11}, x_{21})) + f^3_{trap}(U(x_2, x_{12}, x_{22})) + f^3_{trap}(U(x_3, x_{13}, x_{23})) + \cdots$. In this arrangement, the crossover operator easily disrupts BBs if interactions are not detected correctly. For more details about this test scheme, readers are referred to Goldberg, Korb, and Deb (1989)[2].

Given the failure rate to be 1/10, the population size is set to 182 according to the gambler's ruin model (Harik, Cantú-Paz, Goldberg, & Miller, 1997; Miller, 1997). Binary tournament selection is adopted, and no mutation is used. In the DSM clustering phase, the maximal number of cluster is set to 10, and a bit-wise hill climber is adopted.

[2]In Goldberg, Korb, and Deb (1989), these three cases are referred as tight, loose, and random linkage



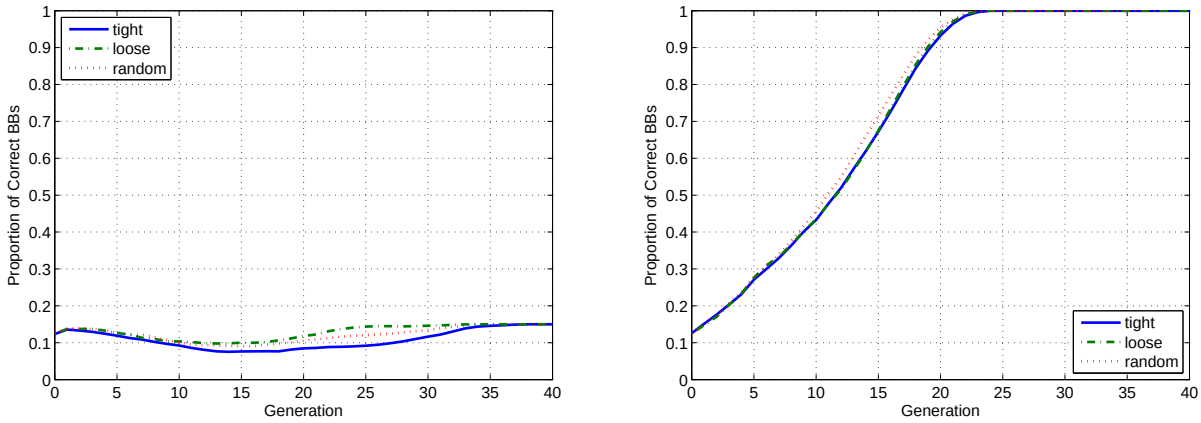
(a) Simple GA with two-point crossover.

(b) DSMGA with BB-wise two-point crossover.

Figure 3.3: The performance comparison of a simple GA with two-point crossover and DSMGA with BB-wise two-point crossover.

Figure 3.3(a) shows the performance of the simple GA using two-point crossover. The simple GA works only for the tight modularity case. For the loose and random modularity cases, the simple GA does not work because of BB disruption. Correspondingly, Figure 3.3(b) illustrates the performance of DSMGA using BB-wise two-point crossover. DSMGA converges for all three tests. Even in the tight modularity test case, DSMGA (converges at the 28^{th} generation) outperforms the simple GA (converges after the 40^{th} generation) because DSMGA disrupts fewer BBs. When the uniform crossover is adopted, the simple GA does not converge in all three test cases. This is because the uniform crossover does not exploit the loci of genes. On the contrary, DSMGA still works on all three test cases. Moreover, DSMGA with the BB-wise uniform crossover works even better than that with BB-wise two-point crossover. It converges roughly at the 22^{th} generation. The reason is that the uniform crossover achieves faster mixing than the two-point crossover.

Figure 3.5 shows the DSM created by DSMGA for the tight modularity case. The perfect result is 10, 3-bit clusters located on the diagonal. At the fifth generation, DSMGA identifies eight BBs correctly; at the tenth generation, DSMGA has successfully identified all ten BBs.



(a) Simple GA with uniform crossover.

(b) DSMGA with BB-wise uniform crossover.

Figure 3.4: The performance comparison of a simple GA with uniform crossover and DSMGA with BB-wise uniform crossover.

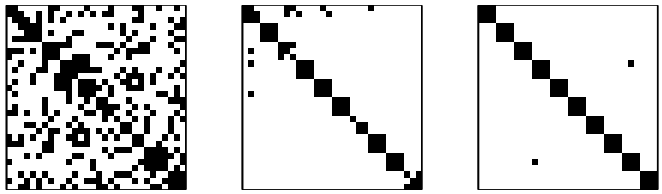


Figure 3.5: The DSMs created by the DSMGA in the tight modularity test. From left to right, the DSMs are created at generation 0, 5, and 10, respectively. The perfect result should be 10, 3-bit clusters on the diagonal.

3.5 Summary and Conclusions

This chapter proposes DSMGA, which utilizes the DSM clustering technique to identify BBs. Three main tasks in DSMGA, DSM construction, DSM clustering, and BB-wise crossover, are described in detail. Empirical results shows that using the BB information obtained by the DSM clustering technique helps the convergence of GAs on problems with tight, loose, and random modularity. The next chapter investigates the scalability of DSMGA.

Chapter 4

Scalability of DSMGA

This chapter analyzes the scalability of DSMGA. Specifically, it proposes a population-sizing model for the entropy-based model building in genetic algorithms (GAs). The effect of the selection pressure on population sizing is also preliminarily incorporated. The proposed model indicates that the population size required for building an accurate model scales as $\Theta(m \log m)$, where m is the number of building blocks (BBs) and is proportional to the problem size. Experiments are conducted to verify the derivations, and the results agree with the proposed model. Given the proposed population-sizing model and the convergence time model developed by Thierens and Goldberg (1994), DSMGA is shown to scale sub-quadratically on boundedly difficult problems both analytically and empirically.

4.1 Population Sizing for Model-building GAs

Genetic evolutionary computation (GEC) researchers have long realized the importance of population sizing on the success and efficiency of GEC. While using a smaller population usually yields low-quality solutions, using a population of size larger than required leads to wasting computational resources. Therefore, facetwise models, such as initial-supply (Goldberg, Sastry, & Latoza, 2001b) and decision-making models (Goldberg, Deb, & Clark, 1992; Harik, Cantú-Paz, Goldberg, & Miller, 1997), have been developed to model different bounds on population sizing required for GA success.

The issue of population sizing is equally critical, if not more, in model-building GAs. One such example is the estimation of distribution algorithms (EDAs) (Larrañaga & Lozano,

2002), which build an interaction model for the given problem and utilize the knowledge gained from the interaction model to efficiently recombine solution candidates. For model-building GAs, the population should be sized properly not only to satisfy the initial supply and the need for making good decisions (Chapter 1), but also to ensure the accuracy of the interaction model.

Pelikan, Sastry, and Goldberg (2003) derived the population size required to build an accurate Bayesian model in BOA to be

$$\Theta(m^{1.05}) \leq n \leq \Theta(m^{2.1}). \tag{4.1}$$

These bounds also apply to many other model-building GAs, and empirical results show that n roughly scales as $\Theta(m^{1.4})$ (Sastry & Goldberg, 2004). However, a more refined model is required to explain the empirical results and better understand population sizing. In addition, empirical results also indicate that selection pressure affects population sizing and that an optimal selection pressure exists for model building.

Many different metrics have been used to detect interactions for model building. One of the most commonly used metrics is Shannon's entropy (Shannon, 1948). Typical examples for such entropy-based model-building GAs include ecGA (Harik, 1999), EBNA (Etxeberria & Larrañaga, 1999), BOA (Pelikan, Goldberg, & Cantú-Paz, 1999), the work of Wright, Poli, Stephens, Landgon, and Pulavarty (2004), and DSMGA.

The purpose of this chapter is to analyze the scalability of DSMGA by developing a facetwise population-sizing model for entropy-based model building in general. The model is anticipated to better explain the scalability of model-building GAs and capture the effect of selection pressure on population-sizing requirements.

The chapter first derives the change of the entropy caused by selection, assuming an infinite population size. Then for a finite population size, the distributions of sampled entropy are investigated. This chapter then shows how selection pressure affects the distributions

and subsequently population sizing. Finally, a population-sizing model based on decision making is derived, and DSMGA scales sub-quadratically to the problem size based on the proposed model.

4.2 Entropy Change Caused by Selection for Infinite Sampling

This section investigates the change of the entropy before and after selection assuming an infinite population size. The loss in entropy by grouping two random variables together defines the mutual information: $\mathbb{I}(X;Y) = \mathbb{H}(X) + \mathbb{H}(Y) - \mathbb{H}(X;Y)$ (Cover & Thomas, 1991). For entropy-based model-building GAs to detect the interaction between genes X and Y, the sampled mutual information between X and Y needs to be significant enough.

The scenario in this chapter is that after unbiased initialization of the population, the GA performs binary tournament selection and then builds the interaction model. The population size required for building an accurate model is then investigated.

The following derivation assumes that the Royal road function (Mitchell, Forrest, & Holland, 1992) is adopted. The Royal road function serves as a worst case scenario for model building in GAs because given the minimal fitness difference d_{min}, the fitness differences between the best schema and all other $(2^k - 1)$ schemata are all d_{min}. In other words, for a fixed d_{min}, the growth of the best schema is the slowest for the Royal road function under a fixed selection pressure. To simplify the derivation, a bipolar Royal road function of order k is defined as follows to restrict the growth rates of 0 and 1 to be the same for every gene.

$$R_k(\vec{x}) = \begin{cases} 1 & \text{if } \vec{x} = \underbrace{111\cdots1}_{k} \\ 1 & \text{if } \vec{x} = \underbrace{000\cdots0}_{k} \\ 1-d & \text{otherwise.} \end{cases} \tag{4.2}$$

The derivation is based on decision making. Similar derivations can be found in the work of Goldberg, Deb, and Clark (1992) and Pelikan, Sastry, and Goldberg (2003). The fitness of an additively decomposable problem with m building blocks (BBs) is then defined as

$$f(\vec{x}) = \Sigma_{i=0}^{m-1} R_k(x_{ik+1}x_{ik+2}\cdots x_{ik+k}). \tag{4.3}$$

Since the number of BBs is proportional to the problem size, for simplicity, the terms "problem size" and "number of BBs" are interchangeable in the context of scalability for the rest of this chapter.

Without loss of generality, the first two genes are chosen to derive the mutual information of two dependent genes. Let X and Y be random variables representing the first and second genes respectively. Also, for a quantity Q before selection, let Q' denote the same quantity after selection. Note that the bipolar Royal road function defined in Equation 4.2 is not biased to 0 or 1. For an infinite population size, at any given gene position, half of the population contains 0 while the other half contains 1 before and after selection. Therefore, the entropy for an individual gene can be calculated as

$$\mathbb{H}(X) = 1, \quad \mathbb{H}(X') = 1, \quad \mathbb{H}(Y) = 1, \quad \text{and} \quad \mathbb{H}(Y') = 1. \tag{4.4}$$

The following derivation calculates the joint entropy of X and Y by investigating the competition between schemata of the first two genes. Define the following notation for schemata

$$H_{xy} = xy\underbrace{***\cdots*}_{l-2}, \tag{4.5}$$

where x and y are 0 or 1, and l is the problem size. Define two sets $H_+ = \{H_{00}, H_{11}\}$ and $H_- = \{H_{01}, H_{10}\}$, and let F_+ and F_- be their fitness values:

$$F_+ = f(H_+) = f(H_{00}) + f(H_{11}), \quad \text{and} \quad F_- = f(H_-) = f(H_{01}) + f(H_{10}). \tag{4.6}$$

According to the central limit theorem (Feller, 1966), the distributions of F_+ and F_- can be approximated as Gaussian distributions when the population size is large. The variances of F_+ and F_-, defined as $\sigma^2_{F_+}$ and $\sigma^2_{F_-}$ respectively, are different but very close. By treating other $(m-1)$ BBs as external noises, these variances can be bounded and approximated as:

$$\begin{aligned} & (m-1)\sigma^2_{BB} \leq \sigma^2_{F_+}, \sigma^2_{F_-} \leq m\sigma^2_{BB} \\ \Rightarrow \quad & \sigma^2_{F_+} \simeq \sigma^2_{F_-} = m\sigma^2_{BB} \cdot (1 - \mathcal{O}(\tfrac{1}{m})), \end{aligned} \tag{4.7}$$

where σ^2_{BB} is the fitness variance of a BB. The difference between those two variances is close to zero and can be neglected when m is large.

Define $Z = F_+ - F_-$. Z is a normally distributed random variable with the following mean and variance.

$$E[Z] = \frac{d}{2^{k-2}}. \tag{4.8}$$

$$Var[Z] = \sigma^2_{F_+} + \sigma^2_{F_-} = 2m\sigma^2_{BB} \cdot (1 - \mathcal{O}(\frac{1}{m})). \tag{4.9}$$

The probability that H_+ wins over H_- in a binary tournament is given by $\Phi(\frac{E[Z]}{\sqrt{Var[Z]}})$, where Φ is the cumulative standard Gaussian distribution. Define a decision variable z as

$$z = \frac{E[Z]}{\sqrt{Var[Z]}} = \frac{d}{2^{k-2}\sqrt{2m} \cdot \sigma_{BB}} + \mathcal{O}(m^{-1.5}). \tag{4.10}$$

For a large m, z is small, and $\Phi(z)$ can be approximated by $\frac{1}{2} + \frac{z}{\sqrt{2\pi}} - \mathcal{O}(z^3)$ (Abramowitz & Stegun, 1970), which yields

$$\Phi(z) = \frac{1}{2} + \frac{d}{2^{k-1}\sqrt{\pi m} \cdot \sigma_{BB}} \pm \mathcal{O}(m^{-1.5}). \tag{4.11}$$

Define p_+ and p_- as the proportions of H_+ and H_- respectively. Before selection, $p_+ =$

$p_- = \frac{1}{2}$. The proportions after selection can be calculated as:

$$p'_+ = \quad p_+^2 + 2p_+p_-\Phi(z) \quad = \frac{1}{2} + \frac{\triangle_m}{4}, \tag{4.12}$$

$$p'_- = \quad p_-^2 + 2p_+p_-\Phi(-z) \quad = \frac{1}{2} - \frac{\triangle_m}{4}, \tag{4.13}$$

$$\text{where} \qquad \triangle_m \quad = \frac{d}{2^k\sqrt{\pi m}\cdot\sigma_{BB}}. \tag{4.14}$$

Equations 4.12 and 4.13 describe the changes of the proportions of H_+ and H_- caused by the selection.

The entropy of a random variable is defined as $\mathbb{H}(\vec{p}) = -\Sigma_i p_i \log_2 p_i$, where p_i is the probability for the i-th event of the random variable to occur. Now calculate the joint entropy of the first two genes before and after selection:

$$\mathbb{H}(X;Y) = \qquad \mathbb{H}(\tfrac{1}{4},\tfrac{1}{4},\tfrac{1}{4},\tfrac{1}{4}) \qquad = 2. \tag{4.15}$$

$$\mathbb{H}(X';Y') = \quad \mathbb{H}(\tfrac{p_+}{2},\tfrac{p_+}{2},\tfrac{p_-}{2},\tfrac{p_-}{2}) \quad = 2 - \frac{\triangle_m^2}{8\ln 2} + \mathcal{O}(\triangle_m^4). \tag{4.16}$$

Combining Equation 4.4, 4.15, and 4.16, the change of the mutual information is given by

$$\mathbb{I}(X';Y') - \mathbb{I}(X;Y) = \frac{\triangle_m^2}{8\ln 2} - \mathcal{O}(\triangle_m^4). \tag{4.17}$$

4.3 Distribution of Entropy for Finite Sampling

This section investigates the effect of selection on the entropy metric for a finite population size. In the case of a finite population, the distribution of the sampled mutual information needs to be considered. Generally speaking, selection increases the sampled mutual information between dependent genes and has no effect on independent genes. Specifically, this section derives the mean and variance of the sampled mutual information.

Define M_0 and M_1 as the mutual information after selection between pairs of independent

genes and dependent genes, respectively. According to the unbiased initialization assumption, $M_0 = 0$, and M_1 is given by Equation 4.17.

Let $\hat{M}_{0,n}$ and $\hat{M}_{1,n}$ be the sampled mutual information for M_0 and M_1 respectively, where n is the sample size. For an infinite number of samples, $E[\hat{M}_{0,\infty}] = M_0 = 0$, and $E[\hat{M}_{1,\infty}] = M_1$. For a finite number of samples, the means and variances of the sampled mutual information can be derived as follows using the expansions in Hutter and Zaffalon (2005).

$$E[\hat{M}_{0,n}] = \frac{1}{2n \ln 2} + \mathcal{O}(\frac{1}{n^2}). \tag{4.18}$$

$$Var[\hat{M}_{0,n}] = \frac{1}{2n^2 \ln 2} + \mathcal{O}(\frac{1}{n^3}). \tag{4.19}$$

$$E[\hat{M}_{1,n}] = \frac{\triangle_m^2}{8 \ln 2} + \frac{1}{2n \ln 2} + \mathcal{O}(\frac{1}{n^2}) - \mathcal{O}(\triangle_m^4). \tag{4.20}$$

$$Var[\hat{M}_{1,n}] = \frac{1}{2n^2 \ln 2} + \frac{\triangle_m^2}{4n \ln 2} - \mathcal{O}(\frac{\triangle_m^2}{n^2}) + \mathcal{O}(\frac{1}{n^3}). \tag{4.21}$$

In Equation 4.21, the first term dominates for a small n while the second term dominates for a large n.

To verify the above derivations, several experiments are conducted by first fixing the problem size and investigating the effect of different population sizes on the sampled mutual information. Equations 4.18 to 4.21 indicate that the sampled mutual information difference, $E[\hat{M}_{1,n} - \hat{M}_{0,n}]$, is virtually independent of n. $Var[\hat{M}_{0,n}]$ is inversely proportional to n^2. $Var[\hat{M}_{1,n}]$ is inversely proportional to n^2 when n is small while inversely proportional to n when n is large. The empirical results shown in Figure 4.1 support the derivation. The experiments are done by applying a GA on the (m, k)-trap (Goldberg, 1987) with m fixed at 10 and $k = 4$. The minimal fitness difference between competing BBs is 0.25. All results are averaged over 10000 independent runs.

Now the effect of different problem sizes on the sampled mutual information is investigated by fixing the population size. Note that $\triangle_m$ is inversely proportional to $\sqrt{m}$ (Equa-

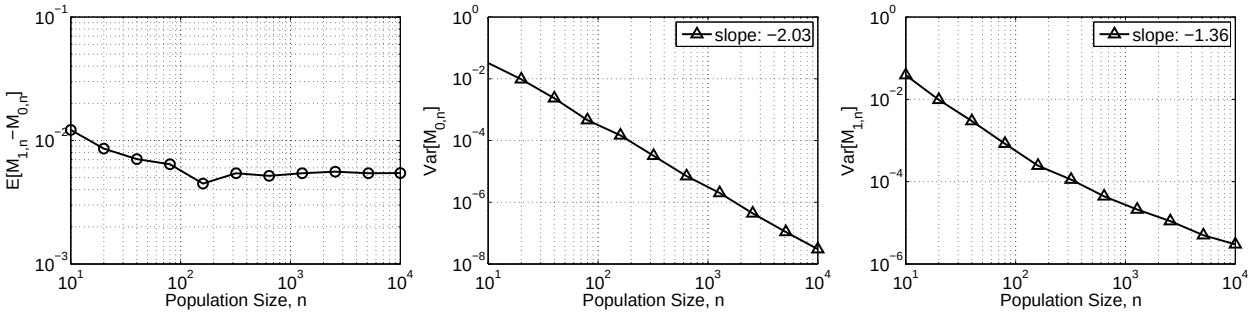


Figure 4.1: The effect of population size on the sampled mutual information. The problem size is fixed. The sampled mutual information difference is virtually independent of n. The sampled information variance for the pair of independent genes roughly scales $\Theta(n^{-2})$. That for the pair dependent genes scales closer to $\Theta(n^{-2})$ for small n and closer to $\Theta(n^{-1})$ for large n.

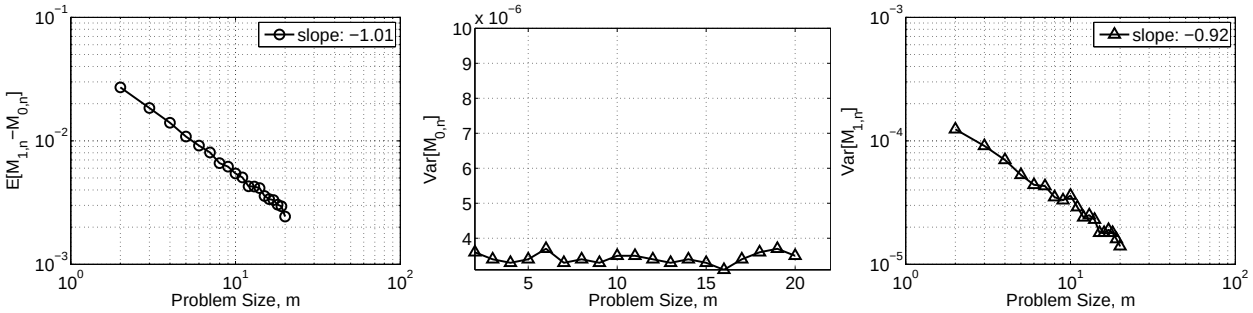


Figure 4.2: The effect of different problem sizes on the sampled mutual information. The population size is fixed. The sampled mutual information difference scales as $\Theta(m^{-1})$. The sampled information variance for the pair of independent genes is virtually independent of m. That for the pair of dependent genes scales as $\Theta(m^{-1})$.

tion 4.14). Neglecting insignificant terms, Equations 4.18 to 4.21 suggest that the difference between the two means, $E[\hat{M}_{1,n} - \hat{M}_{1,n}]$, is inversely proportional to m. Likewise, the variance $Var[\hat{M}_{0,n}]$ is virtually independent of m. $Var[\hat{M}_{1,n}]$ is independent of m when n is small while inversely proportional to m when n is large. Again, the derivation agrees with the empirical results shown in Figure 4.2.

4.4 Effect of Selection Pressure on the Sampled Mutual Information

This section extends the above analysis to tournament selection where the tournament size is s_{to}. While using results from order statistics might accurately capture the effect of selection pressure on population sizing is acknowledged, this section approximates tournament selection by truncation selection to ease the analytical burden.

Consider the scenario where the selection operator is applied multiple times. It provides a similar effect of having an exponentially higher selection pressure. This statement is exact for truncation selection. In truncation selection, selecting the best half of the population twice results in exactly the same population as selecting the best quarter of the population.

Since all derivations in the previous section are based on tournament selection, a transformation from tournament selection to truncation selection is needed. Blickle and Thiele (1995) gave the approximation of the selection intensity for tournament selection as $I = \sqrt{2\left(\ln s_{to} - \ln\sqrt{4.14 \ln s_{to}}\right)}$. On the other hand, Bäck (1995) approximated the selection intensity for truncation selection with a selection pressure s_{tr} as $I = s_{tr}\phi(\Phi^{-1}(1 - \frac{1}{s_{tr}}))$, where ϕ is the probability density function of the standard Gaussian distribution and Φ is the cumulative density function.

By setting the selection intensity to be same, s_{to} and s_{tr} can be solved numerically, and the following relation is obtained. (Figure 4.3).

$$s_{to} \simeq 1.6 s_{tr}. \tag{4.22}$$

Therefore, applying a binary tournament selection has a similar effect as applying truncation selection with a selection pressure $\frac{2}{1.6} \simeq 1.25$. Applying truncation selection t times results in a selection pressure $s_{tr} = 1.25^t$. When t is not too large, Equations 4.12 and 4.13

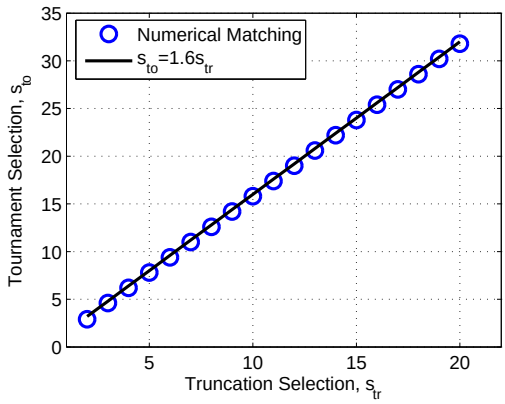


Figure 4.3: The relationship between the tournament size s_{to} and the selection pressure of truncation selection s_{tr} that yield the same selection intensity can be approximated as $s_{to} \simeq 1.6 s_{tr}$.

can be approximately modified as

$$p'_+ = \frac{1}{2} + \frac{t\triangle_m}{4} \qquad \text{and} \qquad p'_- = \frac{1}{2} - \frac{t\triangle_m}{4}. \tag{4.23}$$

As a result, the sampled information for a pair of dependent genes grows proportionally to t^2. On the other hand, the number of samples reduces from n to $\frac{n}{s_{tr}}$ after the selection procedure. Since the work in this chapter focuses on the order of the relationship among population size, problem size, and selection pressure, all constants that are not related to any of these three factors are denoted as c_i for simplicity, where i distinguishes between different constants. Recall that $t = \frac{\ln s_{tr}}{1.25}$ and $s_{tr} = \frac{s_{to}}{1.6}$. The means and the variances of the sampled mutual information can be modeled as:

$$E[\hat{M}_{1,\frac{1.6n}{s_{to}}} - \hat{X}_{0,\frac{1.6n}{s_{to}}}] \simeq c_1 (\ln \frac{s_{to}}{1.6})^2 \triangle_m^2. \tag{4.24}$$

$$Var[\hat{M}_{0,\frac{1.6n}{s_{to}}}] \simeq c_2 \frac{s_{to}^2}{n^2}. \tag{4.25}$$

$$Var[\hat{M}_{1,\frac{1.6n}{s_{to}}}] \simeq c_3 \frac{(\ln \frac{s_{to}}{1.6})^2 s_{tr} \triangle_m^2}{n}. \tag{4.26}$$

Note that the approximation in Equation 4.26 neglects the first term in Equation 4.21,

assuming n is large.

4.5 Population Sizing for Modularity Identification

Chapter 4.3 models the means and variances of the sampled mutual information for the model building on a finite population size. Utilizing these models, this section derives a population-sizing model by the decision-making approach.

According to Hutter and Zaffalon (2005), the distribution of mutual information can be approximated as a Gaussian distribution. The decision-making error can be calculated as follows.

$$\tau \triangleq \frac{E[Z]}{\sqrt{Var[Z]}} \simeq c_4 \frac{\ln \frac{s_{to}}{1.6}}{\sqrt{s_{to}}} \triangle_m \sqrt{n}, \tag{4.27}$$

where $Z = \hat{M}_{1,\frac{1.6n}{s_{to}}} - \hat{M}_{0,\frac{1.6n}{s_{to}}}$. For a large τ, the decision error $\epsilon = 1 - \Phi(\tau)$ can be approximated as

$$\epsilon \simeq \frac{1}{\tau} e^{-\frac{\tau^2}{2}} \simeq \frac{c_5 \sqrt{s_{to}}}{\ln \frac{s_{to}}{1.6} \cdot \triangle_m \sqrt{n}} e^{-c_6 \frac{(\ln \frac{s_{to}}{1.6})^2 \triangle_m^2 n}{s_{to}}}. \tag{4.28}$$

For a problem with m BBs, there are mC_2^k pairs of dependent genes and $C_2^{km} - mC_2^k$ pairs of independent genes. Assuming that genes within a BB are maximally dependent, a BB can be treated as one decision variable, and only C_2^m independent decisions need to be made correctly. Given the model accuracy to be $(1 - \frac{1}{m})$, the following relation holds.

$$(1 - \epsilon)^{\Theta(m^2)} \geq 1 - \frac{1}{m}. \tag{4.29}$$

For a small ϵ and a large m, Inequality 4.29 can be simplified as

$$\begin{aligned} & \tfrac{1}{\epsilon} \geq \Theta(m^3) \\ \Rightarrow \quad & \ln(\tfrac{\ln \frac{s_{to}}{1.6} \cdot \triangle_m \sqrt{n}}{c_5 \sqrt{s_{to}}}) + c_6 \tfrac{(\ln \frac{s_{to}}{1.6})^2 \triangle_m^2 n}{s_{to}} \geq \Theta(\ln m). \end{aligned} \tag{4.30}$$

For a large n, the first term in Equation 4.30 can be neglected. By substituting $\triangle_m$

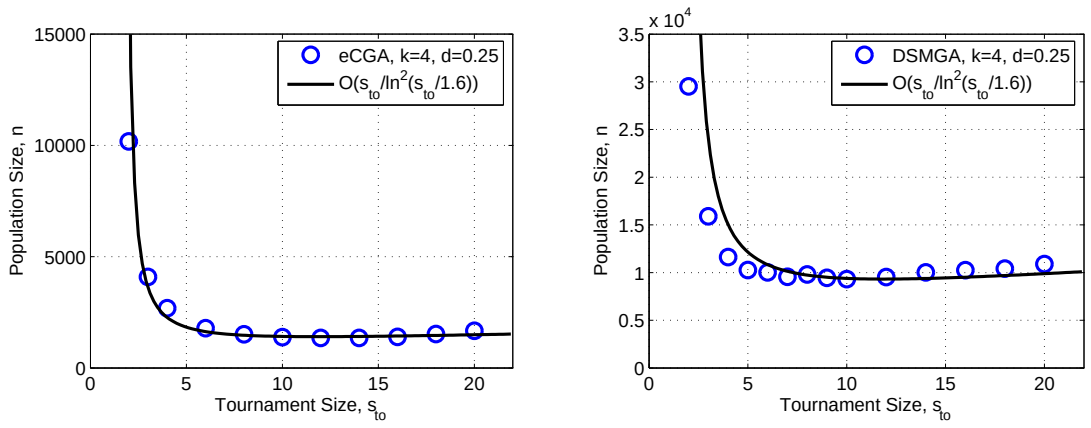


Figure 4.4: The relationship between the tournament size and the population size. Both the results and model indicates the existence of an optimal s_{to} around 10.

according to Equation 4.14, the following bound is obtained.

$$n \geq c_7 \frac{s_{to}}{(\ln \frac{s_{to}}{1.6})^2} 2^{2k} \frac{\sigma_{BB}^2}{d^2} m \ln m. \tag{4.31}$$

The population-sizing model given in Equation 4.31 differs from existing ones in three aspects. First of all, it incorporates the effect of selection pressure. Secondly, it scales as $\Theta(2^{2k})$ instead of $\Theta(2^k)$. Finally, it indicates the population size for model building should be $\Theta(m \ln m)$.

Figure 4.4 shows the relationship between the tournament size and the population size needed to build an accurate model with $(m-1)$ BBs correctly identified. The results agree with the model qualitatively. Basically, for both smaller and larger s_{to}, a larger population size is needed to build an accurate model. Equation 4.31 also predicts a fixed optimal $s_{to}^* \simeq 11.8$. However, empirical results indicate that the optimal tournament size varies with different problems and different model-building procedures. This phenomenon currently is not yet captured in the model. The problem might lie in using truncation selection to approximate tournament selection. Even though the approximation ensures similar selection intensity, in tournament selection, the number of copies of an individual is proportional to its rank, which is not the case for truncation selection.

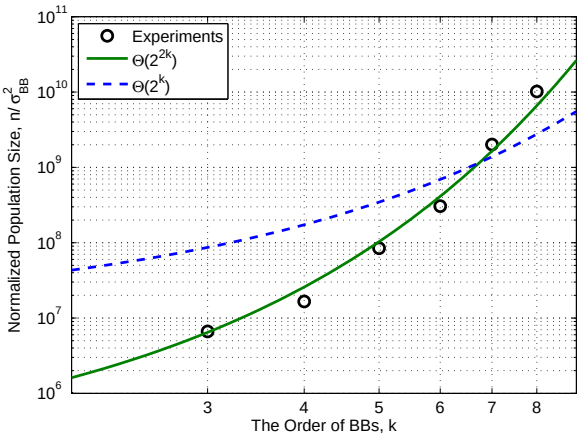


Figure 4.5: The relationship between the population size needed for DSMGA and the order of BBs, k. $\Theta(2^{2k})$ better describes the results than $\Theta(2^k)$ does.

The term 2^{2k} in Equation 4.31 is empirically verified. Figure 4.5 shows the relationship between the population size needed for the model building in DSMGA and the order of BBs, k, for an (m, k)-trap function with $m = 10$. The minimal fitness difference between competing BBs is 0.1. As indicated in the figure, $\Theta(2^{2k})$ better describes the results than $\Theta(2^k)$.

Figure 4.6 shows the experimental results for ecGA and DSMGA on an (m, k)-trap function, where $k = 4$. The fitness difference between competing BBs is 0.25. The power-law curve fitting is done by first-order polynomial fitting on the log-log scale. $\Theta(m \log m)$ provides a better description of the data than the power-law model.

4.6 Summary and Conclusions

This chapter analyzes the scalability of DSMGA by proposing a population-sizing model for entropy-based model building in GAs. Specifically, the population size required for building an accurate model is investigated. The proposed model refines the required population size for model building from $\Theta(m^{1.05}) \leq n \leq \Theta(m^{2.1})$ to $n = \Theta(m \log m)$. It also corrects the term 2^k in existing population-sizing models to 2^{2k}. Those modifications are empirically verified.

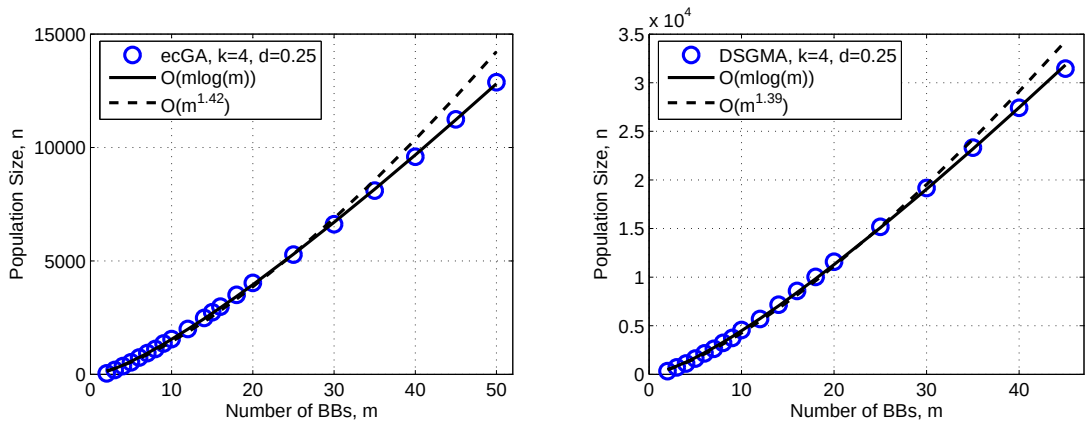


Figure 4.6: The scalability of the population size for different problem sizes. $n = \Theta(m \log m)$ is a better description of the results of both ecGA and DSMGA than the power-law model.

The proposed model also incorporates the effect of selection pressure on the population sizing requirements. Empirical results quantitatively agree well with the proposed model for the scalability on the problem size. The modeling on the selection pressure is qualitatively verified by the results. To obtain a more accurate modeling on the selection pressure, the derivation may need to utilize results from order statistics on the Gaussian distribution.

Compared with the population-sizing model for EDAs (Pelikan, Sastry, & Goldberg, 2003; Sastry & Goldberg, 2004), the proposed population-sizing model scales the same with the problem size as $\Theta(m \log m)$. In other words, the population size required to build a nearly perfect model is of the same order as that needed for GAs to solve the problem, but with a larger constant since GAs do not need a perfect model to solve problems. Also, it is worth noting that the decision-making model (Goldberg, Deb, & Clark, 1992) has a similar form. The difference is that in the decision-making model, decisions are made between competing BBs. Here, decisions are made between pairs of dependent and independent genes. The proposed model indicates that for a low selection pressure, the signal may not be strong enough to detect the existences of interactions; for a high selection pressure, sampling noises may cloud the signal. An optimal selection pressure exists somewhere in the middle for the model builder. Finally, although the proposed model is based on the entropy metric, a

similar procedure should be applied to some other metrics such as nonlinearity (Munetomo & Goldberg, 1999) and simultaneity (Aporntewan & Chongstitvatana, 2003).

With the population-sizing model developed in this chapter and the convergence-time model described in Chapter 1.4, for a problem with m BBs of order k, DSMGA requires $\Theta(2^{2k}m^{1.5}\log m)$ function evaluations to find the global optimal or near-optimal solutions.

Chapter 5

Finding Extrema for Problems with Hierarchy: DSMGA+

So far, it has been shown that DSMGA is capable of solving problems with modularity through proper decomposition. This chapter extends the algorithm to another level of optimization by considering another class of problems–hierarchical problems.

Hierarchical problems come from hierarchical complex system—a system composed of subsystems, and each of which is hierarchical by itself (Simon, 1968). Many complex systems around us are hierarchical. As stated in the introduction, a desktop computer system is composed of software and hardware. Software can be further subdivided into operating systems and applications while hardware can be further subdivided into the core devices and the peripheral devices. Materials are composed of molecules, molecules are composed of atoms, atoms are composed of electrons, protons, neurons, and so forth. A university is composed of colleges, colleges are composed of departments, and so forth.

Inspired by the fact that many complex systems are hierarchical, Pelikan and Goldberg (2001) proposed hBOA, one of few genetic and evolutionary algorithms (Watson & Pollack, 2005) that is known to optimize problems with random modularity by hierarchical decomposition. hBOA has shown the ability to decompose hierarchical problems, which are not fully decomposable in one single level (Pelikan, 2002; Pelikan & Goldberg, 2001).

hBOA has demonstrated an excellent optimization ability; however, the decomposition information is implicitly stored in a Bayesian network, which is usually incomprehensible for human researchers. In many real-world applications, the knowledge of the problem structure is as valuable as a high-quality solution to the problem. For example, in the field of feature selection, one of the most important issues is to discover the dependencies and redundancies

among many features.

The objective of this chapter is to develop an explicit hierarchical decomposition scheme for GAs. The proposed method is based on the dependency structure matrix genetic algorithm (DSMGA) (Yu, Goldberg, Yassine, & Chen, 2003), which utilizes dependency structure matrix clustering techniques for interaction detection. The proposed method optimizes the problem via hierarchical decomposition, and then stores the problem structure in an explicit manner that is transparent to human researchers.

The chapter is organized into four main parts. The first part revisits hierarchical difficulty, hierarchical problems, and the keys to conquer them. The second part describes the proposed explicit chunking: substructural chromosome compression, which reduces the search space on-the-fly. The third part describes the experiments and demonstrates the results. Finally, discussions and some future work conclude this chapter.

5.1 Hierarchical Difficulty and Hierarchical Problems

Hierarchical problems come from hierarchical complex systems. They are not fully decomposable in one single level. In hierarchical problems, the interactions in an upper level are too weak to detect unless all lower levels are solved. This section describes the keys to conquer hierarchical difficulty and details three typical hierarchical problems—the hierarchical IFF, the hierarchical XOR, and the hierarchical trap, which are later used as test functions in the chapter.

5.1.1 Keys to conquer hierarchical difficulty

Pelikan and Goldberg (2001) recognized three keys to conquer hierarchical difficulty.

Proper decomposition. At each level, the problem needs to be properly decomposed so that the GA can mix subsolutions effectively.

Niching. The GA needs to be able to preserve promising subsolutions to the next level because no correct decision can be made until the GA advances to the upper levels.

Chunking. To prevent the complexity of the hierarchical problem from growing exponentially, the GA needs to represent one block in a lower level as one variable in an upper level.

In hBOA, the chunking is implicit and is achieved by recognizing local substructures in the Bayesian network. This chapter proposes an explicit chunking scheme, which is more comprehensible. Details will be described in Chapter 5.2.

5.1.2 The design of hierarchical problems

Several hierarchical problems were designed to test the methodology. The design is guided by the three keys described in the previous section: (1) proper decomposition, (2) niching, and (3) chunking. Three test problems used in this chapter are the hierarchical IFF (Watson, Hornby, & Pollack, 1998), the hierarchical XOR (Watson & Pollack, 1999), and the hierarchical trap (Pelikan & Goldberg, 2001). They are detailed as follows.

Hierarchical if and only if (hIFF)

The hierarchical IFF problem is defined on a binary string $x \in \{0,1\}^{2^\lambda}$, where λ is the number of hierarchical levels. First, define a Boolean function $h(x)$ to determine if x is *valid* or not. Let $L = x_1 x_2 \cdots x_{2^{\lambda-1}}$, and $R = x_{2^{\lambda-1}+1} x_{2^{\lambda-1}+2} \cdots x_{2^\lambda}$.

$$h_{iff}(x) = \begin{cases} 1 & \text{if } \lambda = 0 \\ 1 & \text{if } h_{iff}(L) = 1, h_{iff}(R) = 1, \text{ and } L = R \\ 0 & \text{otherwise.} \end{cases} \tag{5.1}$$

Based on $h_{iff}(x)$, for $\lambda > 0$, the fitness of hIFF is defined recursively as:

$$H_{iff}(x) = H_{iff}(L) + H_{iff}(R) + \begin{cases} \text{length}(x) & \text{if } h_{iff}(x) = 1 \\ 0 & \text{otherwise.} \end{cases} \tag{5.2}$$

The base case is when $\lambda = 0$, $H_{iff}(x) = 1$. hIFF has two global optima: a string with all 0's and with all 1's. hIFF has $2^{l/2}$ local optima at the lowest level for a problem size l.

Hierarchical exclusive or (hXOR)

The global optima of hIFF are all 1's and all 0's, which might not be too hard to find for an algorithm biased or drifted to some particular allele value. To prevent the search algorithm from exploiting this problem property, the hierarchical XOR problem is designed.

The definition of hXOR is very similar to that of hIFF, but with an alternation in the validation function by the complement check.

$$h_{xor}(x) = \begin{cases} 1 & \text{if } \lambda = 0 \\ 1 & \text{if } h_{xor}(L) = 1, h_{xor}(R) = 1, \text{ and } L = \overline{R} \\ 0 & \text{otherwise,} \end{cases} \tag{5.3}$$

where $\overline{R}$ is the bitwise negation of R. Similarly, for $\lambda > 0$, the fitness of hXOR is defined recursively as:

$$H_{xor}(x) = H_{xor}(L) + H_{xor}(R) + \begin{cases} \text{length}(x) & \text{if } h_{xor}(x) = 1 \\ 0 & \text{otherwise.} \end{cases} \tag{5.4}$$

The base case is when $\lambda = 0$, $H_{xor}(x) = 1$. Similar to hIFF, hXOR also has two global optima and $2^{l/2}$ local optima at the lowest level for a problem size l. However, there are exactly half of 1's and half of 0's in the global optima. Table 5.1.2 lists one of the global

level	global optimum	# of optima
0	1	2
1	10	2
2	1001	4
3	10010110	16
4	1001011001101001	256
5	10010110011010010110100110010110	65536

Table 5.1: One of the global optima and the number of optima of hXOR for level 0 to 5. There are half 0's and 1's in the global optima, and the number of optima grows exponentially with the problem size.

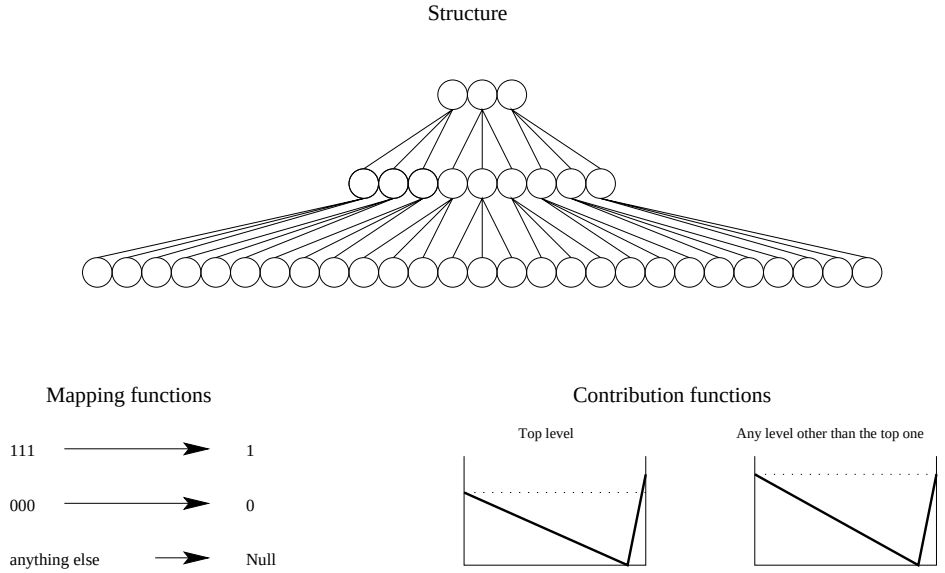


Figure 5.1: The structure, mapping function, and contribution functions of the hierarchical trap. This example has three levels and $k = 3$. Note that 111 and 000 are equally good except for the top level.

optima for level 0 to 5.

Hierarchical trap (hTrap)

Pelikan and Goldberg (2001) proposed a scalable hierarchical problem called the *hierarchical trap* (hTrap). hTrap is composed of three major components (Figure 5.1):

1. **Structure.** The hierarchical trap structure is a balanced k-ary tree.

2. **Mapping function.** The mapping function maps genes from lower levels to upper

levels. A block of all 0's and 1's is mapped to 0 and 1 respectively, and everything else is mapped to '-'.

3. **Contribution functions.** The contribution function is based on trap functions of order k. There are two parameters in the trap functions: f_{high} and f_{low}, which control the degree of deception. The trap function is defined as:

$$trap_k(u) = \begin{cases} f_{high} & \text{if } u = k \\ f_{low} \times \frac{k-1-u}{k-1} & \text{otherwise,} \end{cases} \tag{5.5}$$

where u is the unitary of the input string. If any position of the string is '-', the contribution of the string to the fitness is zero.

The hierarchical trap functions used in this chapter have k set to 3. Both f_{high} and f_{low} are set to 1 for all but the highest level. In the highest level, $f_{high} = 1$ and $f_{low} = 0.9$. So the decision between competing BBs cannot be made correctly until the GA reaches the highest level.

5.2 Substructural Chromosome Compression

This section proposes an explicit chunking scheme, named the *substructural chromosome compression*. The substructural chromosome compression scheme expresses a building block (BB) by one single variable when the BB nearly converges to only few expressive schemata.

The idea of using standard text compression techniques in GAs has been explored elsewhere (Toussaint, 2005), but the method to date only works with explicit prior knowledge of model boundaries, an assumption that makes the technique virtually unusable in most applications. Nevertheless, the idea of compression is sound and the goal of this section is to realize a practical and broadly applicable technique to solve problems on a larger scale.

The key idea of substructural chromosome compression is to represent a nearly-converged BB by only π of the most expressive schemata. Gambler's ruin population-sizing model (Harik, Cantú-Paz, Goldberg, & Miller, 1997) estimates the population size to be $O(2^k m)$ for a binary problem with m BBs of order k. In a typical hierarchical problem, the second level has (m/k) BBs with order k. Since the complexity of a higher level is expected to decrease, the following condition should hold.

$$\pi^k \left(\frac{m}{k}\right) \leq 2^k m, \tag{5.6}$$

or simply

$$\pi \leq 2\sqrt[k]{k}. \tag{5.7}$$

Note that π decreases when k increases, and for $k \geq 3$, π is less than 3. Therefore this thesis only focuses on the case where $\pi = 2$ although the method is not limited to this special case. In other words, when a BB nearly converges, it is compressed to one single bit, where 1 maps to the most expressive schema and 0 maps to the second most expressive schema. The information of the other $(2^k - 2)$ less expressive schemata are discarded.

One thing very critical for the substructural chromosome compression scheme is that every BB of lower levels needs to be compressed **before** the GA can advance to an upper level. Otherwise, the number of schemata in a BB would be 2^{k^2} in the next level, and the GA fails because of insufficient BB supply. To avoid this situation from happening, two things need to be taken care of: (1) compress a BB as soon as the decision can be made with high confidence, and (2) make sure that interaction of an upper level is not expressed before all BBs of the lower level are compressed. The methods are described in detail in following subsections.

5.2.1 Compression criterion

As mentioned before, a BB is compressed to one single bit when it is nearly converged. Neglecting the possibility that the most expressive schema in the current population could be wrong, the following derivation is based on making a good decision between the second (H_2) and the third (H_3) most expressive schema. Note that the derivation is similar to that of the decision-making population-sizing model (Goldberg, Deb, & Clark, 1992).

Assume that the proportions of H_2 and H_3 in the current population are p_2 and p_3, respectively, and by definition, $p_2 \geq p_3$. If the GA reaches a steady state, the proportions are of binomial distribution with n samples, where n is the population size. The probability of making an error may be calculated as:

$$\epsilon = 1 - \Phi(\frac{p_2 - p_3}{\sqrt{\sigma_{H_2}^2 + \sigma_{H_3}^2}}), \tag{5.8}$$

where Φ is the standard cumulative Gaussian function and $\sigma_{H_2}^2$ and $\sigma_{H_3}^2$ are variances of p_2 and p_3, respectively. Assuming steady-state, the variances can be approximated as $\sigma_{H_2}^2 \simeq \frac{p_2(1-p_2)}{n}$, and similarly, $\sigma_{H_3}^2 \simeq \frac{p_3(1-p_3)}{n}$.

Since there is no turning back once a BB is compressed, the room for the decision-making error is small. Here the derivation adopts an error tolerance similar to the GA convergence condition: For a problem with m BBs, at most one BB can be wrong. Define a decision variable z by:

$$z = \frac{p_2 - p_3}{\sqrt{\sigma_{H_2}^2 + \sigma_{H_3}^2}}. \tag{5.9}$$

For $z \gg 1$, the error can be approximated as

$$\epsilon \simeq \frac{1}{\sqrt{2\pi} z e^{\frac{z^2}{2}}}. \tag{5.10}$$

Since at most one error can occur among all m BBs, the following relation holds.

$$(1-\epsilon)^m \geq \frac{m-1}{m}. \tag{5.11}$$

For a large m, the above inequality can be approximated as $\epsilon \leq \frac{1}{m^2}$, which yields

$$\frac{1}{\sqrt{2\pi}ze^{\frac{z^2}{2}}} \leq \frac{1}{m^2}. \tag{5.12}$$

The above inequality can be solved as:

$$z \geq \sqrt{\mathbb{W}(\frac{m^4}{2\pi})}, \tag{5.13}$$

where $\mathbb{W}$ is Lambert's W-function. For a problem with unknown m, conservatively assuming $m = l$ yields $z = \sqrt{\mathbb{W}(\frac{l^4}{2\pi})}$.

Finally, a BB is compressed when the signal difference satisfies

$$p_2 - p_3 \geq z\sqrt{\sigma_{H_2}^2 + \sigma_{H_3}^2}, \tag{5.14}$$

where $\sigma_{H_2}^2$ and $\sigma_{H_3}^2$ can be approximated by the binomial distribution as $\frac{p_2(1-p_2)}{n}$ and $\frac{p_3(1-p_3)}{n}$, respectively, given the population size of n.

After the interaction information is retrieved from the DSM clustering, the GA checks if Inequality 5.14 is satisfied for every BB. If so, the BB is compressed to one single bit by converting the most expressive schema to 1, the second most expressive schema to 0, and any other schemata randomly to 1 or 0.

5.2.2 Interaction detection thresholds

Now a proper threshold is needed to prevent the upper-level interactions from expressing before all BBs in the lower level are compressed.

In Chapter 3.3, a threshold for noise resistance is given as:

$$\theta_1 = \frac{1}{2n} + \sqrt{\mathbb{W}(\frac{1}{8\pi k^2} l^6) \frac{1}{2n^2}}. \tag{5.15}$$

If the sampled mutual information of two genes is less than θ_1, these two genes are considered to be independent, and the corresponding DSM entry is 0.

Now consider those sampled mutual information that are greater than θ_1. Another threshold is required to separate the interactions of the current lowest level from that of all other upper levels. For a hierarchical problem, the strengths of the interactions are stronger in lowers levels and weaker in upper levels. The task here is equivalent to finding the cluster with the greatest mean in a set of numbers. Since the signal strength of the interactions from upper levels are significantly weaker than that from the current lowest level, a simple k-mean clustering algorithm where $k = 2$ should be able to distinguish these interactions. The splitting point from the k-mean algorithm gives the desired threshold θ_2.

Now that both thresholds, θ_1 and θ_2, are computed, the DSM can be constructed as $DSM = [d_{ij}]$ where

$$d_{ij} = \begin{cases} 1 & \text{if } I(\text{gene}_i; \text{gene}_j) > \max(\theta_1, \theta_2). \\ 0 & \text{otherwise.} \end{cases} \tag{5.16}$$

The DSM constructed above should mainly contain the dependency information from the current lowest level, which has the strongest dependencies by definition.

5.2.3 Putting it all together: DSMGA+

Recall that to conquer the hierarchical difficulty, a GA needs to maintain (1) proper decomposition at each level, (2) preservation of alternative solutions, and (3) representation of a chunk at the lower level as one single variable at the upper level. Now that all tools are

	Mean	Standard Deviation
Level 1 Dependency	0.09064	0.02200
Level 2 Dependency	0.00291	0.00355
Level 3 Dependency	0.00151	0.00212
Threshold (θ_2)	0.04543391	

Table 5.2: The second threshold computed from the k-mean algorithm and the means and variances of the measured mutual information from different levels of a 3-level hTrap with $k = 3$ at the first generation. The algorithm is able to distinguish level-1 dependency from dependencies of other levels.

available for the three key points, a GA that solves problems via hierarchical decomposition, named DSMGA+, is developed on the basis of DSMGA (Yu, Goldberg, Yassine, & Chen, 2003).

DSMGA+ utilizes DSM clustering techniques to decompose the hierarchical problem at each level. It adopts restricted tournament replacement (RTR) (Harik, 1994; Pelikan & Goldberg, 2001) to preserve promising subsolutions just like hBOA does. Unlike hBOA, however, DSMGA+ takes the advantage of the explicit interaction model and achieve an explicit chunking scheme, the substructural chromosome compression.

To conclude, the substructural chromosome compression scheme expresses a building block by one single bit when the alleles in the building block nearly converge to only two niches. It reduces the problem complexity on the fly, and shrinks the search space when parts of the problem are solved.

5.3 Empirical Results

First, the k-mean clustering algorithm is performed to determine whether the desired threshold can be found. The test function is a 3-level hTrap function with $k = 3$, and population size n is set to 2000. The sampled mutual information should come from three different distributions (level-1, level-2, and level-3 dependencies). Table 5.2 shows the means and variances of the measured mutual information from three different levels as well as the sec-

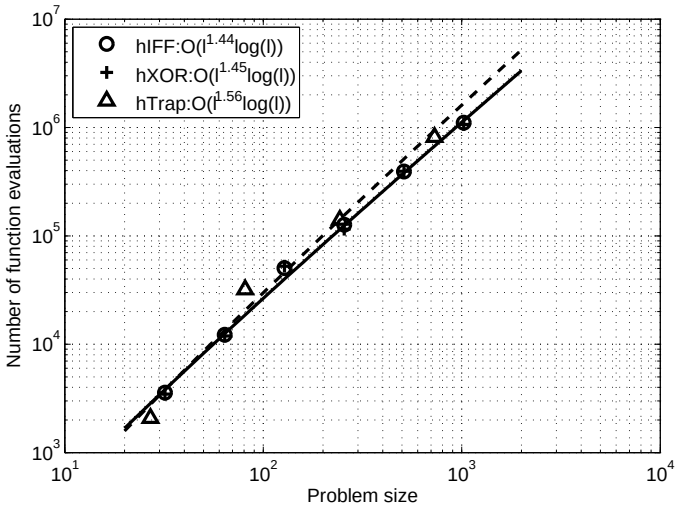
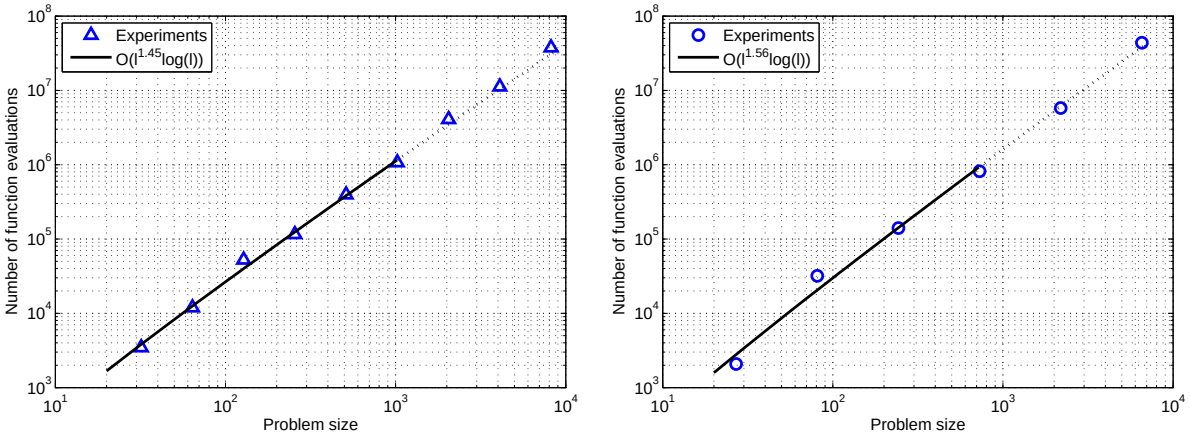


Figure 5.2: Scalability of DSMGA+ on hIFF, hXOR, and hTrap. The number of function evaluations of DSMGA+ scales as $O(l^{1.44}\log(l))$ on hIFF, $O(l^{1.45}\log(l))$ on hXOR, and $O(l^{1.56}\log(l))$ on hTrap, where l is the problem size.

ond threshold (θ_2) that the k-mean clustering algorithm computes. As the results indicate, the algorithm can distinguish the level-1 dependency from dependencies of other levels.

The scalability of DSMGA+ is tested on hIFF, hXOR, and hTrap with results shown in Figure 5.2. The number of function evaluations of DSMGA+ scales as $O(l^{1.44}\log(l))$ on hIFF, $O(l^{1.45}\log(l))$ on hXOR, and $O(l^{1.56}\log(l))$ on hTrap, where l is the problem size. The scaling orders are similar to that of hBOA (Pelikan, 2002) and show that DSMGA+ scales sub-quadratically on these hierarchical problems. The results well indicate that the proposed substructural chromosome compression scheme functions as expected since it is known that the number of function evaluations scales exponentially to the problem size without a proper chunking mechanism (Pelikan, 2002). Figure 5.3 shows the scalability test extended to problem sizes close to ten thousand. Since DSMGA+ produces almost identical results on hIFF and hXOR, the test is performed only on hXOR and hTrap. Due to limited computational resource, for hXOR with problem size larger than 1024, DSMGA+ is performed only once on an extrapolated population size according to the results for smaller problems. Same guideline applies to hTrap with problem size larger than 729.

Theoretically, the substructural chromosome compression helps DSMGA+ solve hierar-



(a) hXOR (b) hTrap

Figure 5.3: Scalability of DSMGA+ on hXOR and hTrap with problem size close to 10^4. DSMGA+ is performed only once for those data points on the dashed lines.

chical problem in a hierarchical manner. At the beginning, DSMGA+ only observes the interactions of the lowest level. Therefore, the hierarchical problem is considered as a one-level problem with modularity. Take hTrap as an example, for $k = 3$ and $l = 27$, DSMGA+ views the problem as a (m, k)-trap with $m = 9$ and $k = 3$ at the beginning. After every first-level BB is compressed, DSMGA+ views the problem as another (m, k)-trap with $m = 3$ and $k = 3$. Chapter 4 indicates that the number of function evaluations needed for DSMGA to solve a problem with m modules of order k scales as $\Theta(2^k m^{1.5} \log m)$. Therefore, for a hierarchical problem of size l with a constant order of modules k, the number of function evaluations needed should scale as $n_{fe} = \Theta(2^k \frac{l}{k}^{1.5} \log \frac{l}{k}) + \Theta(2^k \frac{l}{k^2}^{1.5} \log \frac{l}{k^2}) + \cdots = \Theta(2^k \frac{l}{k}^{1.5} \log \frac{l}{k})$. Given that k is constant, n_{fe} scales as $\Theta(l^{1.5} \log l)$. This analysis agrees with the empirical results.

As mentioned, one of the advantages of DSMGA+ over hBOA is that DSMGA+ delivers the problem structure in a comprehensible manner to human researchers. The problem structures obtained by DSMGA+ for hXOR and hTrap are shown in Figures 5.4 and 5.5 respectively. The problem structure obtained from DSMGA+ on hIFF is very similar to that on hXOR, and hence is omitted. The tested hXOR has 4 levels, and the chromosome

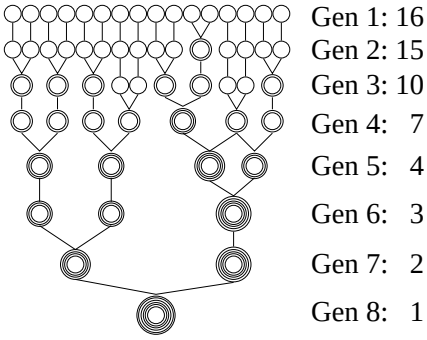


Figure 5.4: Problem structure obtained by DSMGA+ for the hXOR function with 4 levels (chromosome length= $2^4 = 16$). The nodes represent genes of chromosomes. The number of circles represents the compression level. The descriptions on the right show the generation number and the chromosome length. The ideal case would be a complete binary tree with 4 layers. The problem structure obtained by DSMGA+ for hIFF is very similar to this.

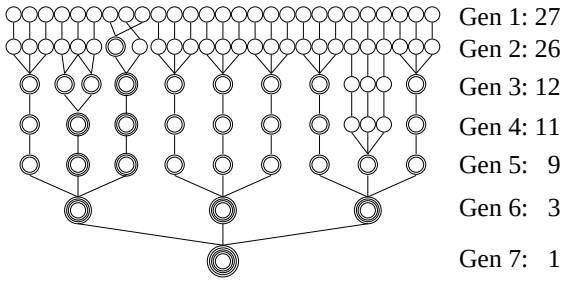


Figure 5.5: Problem structure obtained by DSMGA+ for the hTrap function with 3 levels and $k = 3$ (chromosome length= $3^3 = 27$). The nodes represents genes of chromosomes. The number of circles represents the compression level. The descriptions on the right show the generation number and the chromosome length. The ideal case would be a complete 3-ary tree with 3 layers (Figure 5.1).

length is $2^4 = 16$. The tested hTrap has 3 levels, and the chromosome length is $3^3 = 27$. The perfect problem structure would be a complete binary tree with 4 layers for hXOR and a complete 3-ary tree with 3 layers for hTrap. The figure gives a few indications: (1) that sometimes some genes in a BB are clustered together before other genes join the cluster, (2) that some BBs converge faster than others, and (3) that sometimes overlaps occur. These imperfections are due to sampling noises. Nevertheless, DSMGA+ still captures the problem structures pretty well.

Figure 5.6 illustrates the behavior of the substructural chromosome compression scheme on hIFF, hXOR, and hTrap. The tested hIFF and hXOR have 6 levels and chromosome

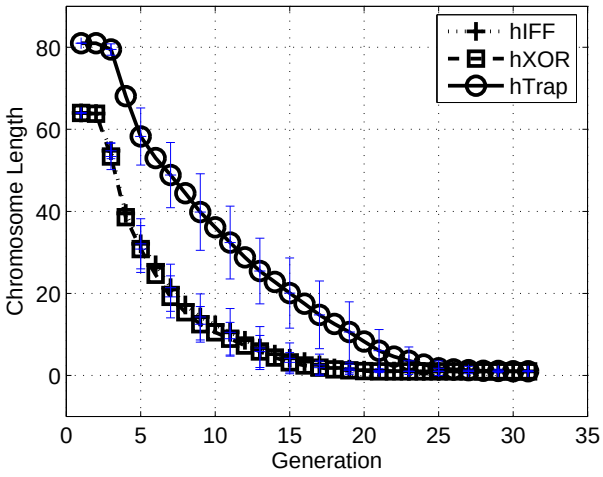


Figure 5.6: Chromosome length reduction for a level-6 hIFF, a level-6 hXOR, and a level-4, order-3 hTrap. The results are averaged over 100 independent runs.

length of $2^6 = 64$. The tested hTrap has 4 levels and chromosome length of $3^4 = 81$. The results are averaged over 100 independent runs. It can be seen that the chromosome lengths keep reducing during the GA runs, which helps the GA to solve the problem more efficiently.

5.4 Summary and Conclusions

This chapter extends DSMGA to optimize problems via hierarchical decomposition. DSMGA+ utilizes DSMGA to decompose the problem at each level, restricted tournament replacement (RTR) to preserve alternative promising subsolutions, and a proposed substructural chromosome compression scheme to represent subsolutions at lower levels as decision variables at upper levels.

The scalability test indicates that DSMGA+ scales sub-quadratically on the tested hierarchical problems, including hIFF, hXOR, and hTrap. It also demonstrates that DSMGA+ is capable of capturing the problem structures for those test functions.

Two points about DSMGA+ should be under the spotlight. (1) DSMGA+ is capable of solving problems via hierarchical decomposition, and it scales sub-quadratically with the problem size; (2) once the problem is solved, DSMGA+ delivers the problem structure as

well. Compared to hBOA, the optimization procedure of DSMGA+ is more transparent to human researchers. The way that DSMGA+ optimizes problems is to automatically create a customized recombination operator for the problem. After applying DSMGA+ to a small-scale problem, one should be able to apply the customized recombination operator to a larger-scale problem of similar structure without the expense of model building.

As for future work, it would be interesting to apply DSMGA+ to real-world problems and demonstrate the reusability of the recombination operator constructed by DSMGA+.

Chapter 6

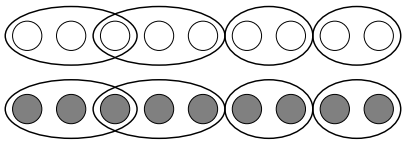
Preparation for Overlap Difficulty: Two Errors in Model Building

Chapter 3 demonstrates how to efficiently solve problems with modularity by recombining subsolutions to subproblems. The proposed DSM clustering technique is capable of identifying overlapping modules. However, the recombination of overlapping subsolutions needs to be done carefully; otherwise, more harm than good may occur.

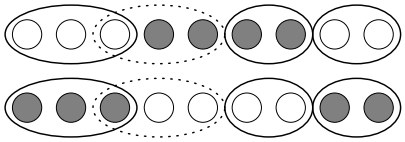
This chapter discusses how to achieve efficient recombination of subsolutions for problems with overlap by investigating the tradeoff between two types of model errors. Consider a problem composed of two overlapping subproblems and other non-overlapping subproblems. For those non-overlapping subproblems, BB-wise uniform crossover gives a good mixing. Two choices can be made when dealing with the overlapping subproblems (Figure 6.1). The first choice would be to mix the two overlapping subsolutions and hence disrupts one of them. The other choice treats the two overlapping subproblems as one bigger subproblem and does not disrupt the subsolutions at all; however, due to no decomposition, the complexity of the problem is not reduced.

In the context of GAs, mixing overlapping subsolutions causes BB disruptions while treating overlapping subproblems as one larger subproblem slows BB mixing. To make the decision wisely, a better understanding of how BB disruption and BB mixing affect the run duration of GAs is needed. This chapter tempts to develop such facetwise models to guide the recombination for problems with overlap.

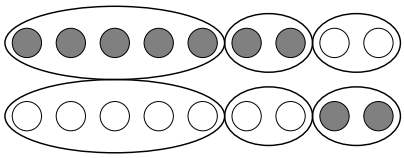
This chapter starts by recognizing two types of errors that could occur in an interaction model. Subsequently, qualitative and quantitative facetwise models are derived to describe the effects of these errors on the GA convergence time. This chapter concludes by recog-



(a) A problem with two overlapping BBs and two non-overlapping BBs.



(b) Breaking one overlapping BB.



(c) Treating two overlapping BBs as one big BB.

Figure 6.1: Two possible ways to recombine two chromosomes with overlapping BBs.

nizing three keys for efficient and effective mixing: (1) minimization of BB disruptions, (2) maximization of effective exchange length, and (3) nondeterminism of information exchange.

6.1 Assumptions and Two Types of Errors in Interaction Models

In this chapter, several assumptions are made to simplify the derivations:

Selectorecombinative GAs: In this chapter, the crossover probability is always 1, and mutation is not taken into consideration. In other words, this chapter focuses only on the mixing behavior of the crossover operator.

Fixed-length χ-ary encoding: The chromosome length is assumed to be fixed. The results can be applied to non-binary encoding, but the size of alphabets should be bounded.

Infinite population size: This is a necessary assumption to use the convergence-time model derived in Mühlenbein and Schlierkamp-Voosen (1993) and Thierens and Goldberg (1994). To mimic the asymptotic behavior, a population size of 10 times what the

gambler's ruin model (Harik, Cantú-Paz, Goldberg, & Miller, 1997) predicts is used for all experiments. For finite population correction, readers are referred to Rattray and Shapiro (1997).

BB-wise uniform crossover: The BB-wise uniform crossover is similar to allele-wise uniform crossover, except using BBs as operands instead of alleles. The BB information is retrieved by some model-building algorithm and may not be necessarily accurate.

An interaction model tells which genes form BBs and comes in different forms, depending on the associated GA. For instance, the Boolean flags in LEGO (Smith & Fogarty, 1996), the genetic ordering in LLGA (Harik, 1997), the clustering model in ecGA (Harik, 1999), and the DSM clustering arrangement in DSMGA (Yu, Goldberg, Yassine, & Chen, 2003) are all interaction models.

Two types of errors can happen when an interaction model describes genetic interactions. One is that the interaction model does not link those genes that are linked in reality, called *detection failure*. The other is that the interaction model links those genes that are not linked in reality, called *false interaction*. The quality of an interaction model can be quantified by the number of errors it makes. For example, consider a problem with four BBs, where $\{BB_1$, BB_2, BB_3, $BB_4\}$=$\{\{1,2,3\}$, $\{4,5,6\}$, $\{7,8,9\}$, $\{10,11,12\}\}$, and an interaction model $\{BB'_1$, BB'_2 ,BB'_3, BB'_4, $BB'_5\}$=$\{\{1,2\}$, $\{3,4,5,6\}$, $\{7,8\}$, $\{9,10,11\}$, $\{12\}\}$. The interaction model has 3 detection-failure errors and 1 false-interaction error.

6.2 Detection Failure and Convergence Time

This section focuses on the detection-failure errors in a interaction model and investigates the errors' relationship to GA convergence time.

6.2.1 Building block disruptions

According to Goldberg et al. (1992), mixing good BBs is critical for GA success. In most traditional GAs, BB mixing is done by performing crossover. However, if BBs are not correctly identified, crossover also disrupts BBs. This subsection derives an upper bound for the expected number of BB disruptions given the number of detection-failure errors in the interaction model.

Define two terms: *correct* BB and *incorrect* BB. If the genes in a BB have the same values as those genes at the same locations of the globally optimal solution, the BB is called a correct BB; otherwise, it is called an incorrect BB. A BB disruption occurs when a correct BB becomes an incorrect BB after crossover.

By assumption, during crossover, a misidentified BB is recombined by portions coming from two different BBs. When one portion comes from a correct BB and the other comes from an incorrect BB, a BB disruption probably occurs. To be conservative, assume that such recombination always results in BB disruptions. This happens when the most competitive incorrect BB is exactly the compliment of the correct BB. Likewise, assume that recombining incorrect BBs always produces incorrect BBs.

Assume that the current population contains a proportion of p correct BBs. For a randomly chosen BB, a BB disruption occurs when (1) it is misidentified, (2) it is a correct BB, and (3) it is going to be recombined with an incorrect BB. Therefore, the probability of a BB disruption occurrence is given by

$$\left(1-(1-\frac{1}{m})^{e_d}\right)p(1-p), \tag{6.1}$$

where m is the number of BBs in a chromosome and e_d is the number of detection-failure errors. When e_d is much less than m, the above equation can be approximated as

$$\frac{e_d}{m}p(1-p). \tag{6.2}$$

A population of size n contains nm BBs in total. The expected number of BB disruptions is then $nm \times \frac{e_d}{m} p(1-p) = ne_d p(1-p)$.

6.2.2 Time-to-convergence model

Recall that the time-to-convergence for GAs on a problem with m uniformly scaled BBs can be modeled as (Mühlenbein & Schlierkamp-Voosen, 1993; Thierens & Goldberg, 1994; Miller, 1997) (Chapter 1.4):

$$p_{t+1} - p_t = \frac{I}{\sqrt{m}} \sqrt{p_t(1-p_t)}, \tag{6.3}$$

where I is the selection pressure and p_t is the proportion of correct BBs at generation t.

When the interaction model has some detection-failure errors, the growth of correct BBs slows down due to BB disruptions. After selection, the growth of correct BBs is still governed by Equation 6.3, $p_{t,selected} = p_t + \frac{I}{\sqrt{m}} \sqrt{p_t(1-p_t)}$. The proportion of disrupted BBs is given by Equation 6.2 as $\frac{e_d}{m} p_{t,selected}(1 - p_{t,selected})$. The proportion of correct BBs for the next generation can then be calculated by subtracting BB disruptions from BB growth:

$$p_{t+1} = p_{t,selected} - \frac{e_d}{m} p_{t,selected}(1 - p_{t,selected}). \tag{6.4}$$

By approximating the BB disruption in Equation 6.4 as $\frac{e_d}{m} p_t(1-p_t)$ (the proportion of disrupted BBs is calculated according to the proportion of correct BBs before selection) and adopting $p(1-p) \leq \frac{1}{2}\sqrt{p(1-p)}$ for $0 \leq p \leq 1$, the proportion of disrupted BBs can be approximately upper bounded by $\frac{e}{2m} \sqrt{p_t(1-p_t)}$. To be conservative, assume the above upper bound estimates the proportion of disrupted BBs. The proportion of correct BBs for the next generation is then given by

$$p_{t+1} - p_t = (\frac{I}{\sqrt{m}} - \frac{e_d}{2m}) \sqrt{p_t(1-p_t)}. \tag{6.5}$$

Following a similar procedure in the work of Mühlenbein and Schlierkamp-Voosen (1993) and Thierens and Goldberg (1994) (Chapter 1.4), one can derive the time-to-convergence as a function of e_d.

$$t_{conv}(e_d) = \frac{\left(\frac{\pi}{2} - \sin^{-1}(2p_0 - 1)\right)}{\frac{I}{\sqrt{m}} - \frac{e_d}{2m}}. \tag{6.6}$$

The dimensionless model can be obtained as

$$\frac{t_{conv}(0)}{t_{conv}(e_d)} = 1 - \frac{e_d}{2I\sqrt{m}}. \tag{6.7}$$

The above equation also suggests that when $e_d \geq 2I\sqrt{m}$, BB disruptions outweigh BB growth; the GA then acts like a random search and is difficult to converge. Define the *critical number of errors*, $e_{critial}$, as the maximal number of BBs that an interaction model could misidentify while a $(m-1)$-BB convergence is still possible. The relation between $e_{critial}$ and m can be expressed as

$$e_{critial} = 2I\sqrt{m}. \tag{6.8}$$

Since I is only loosely related to the problem, the key idea of the Equation 6.8 is that $e_{critial} = \Theta(\sqrt{m})$.

6.2.3 Empirical Results

The derived models are empirically verified. The test function is a (m, k)-trap, where $m = 100$ and $k = 5$. To approximate the asymptotic behavior of the time-to-convergence model, the population is sized as four times of what the gambler's ruin model estimates (Harik, Cantú-Paz, Goldberg, & Miller, 1997).

The relationship between the number of BB disruptions and the correct BB proportion for different numbers of detection-failure errors is presented in Figure 6.2. The figure shows that the derived model (Equation 6.4) is more accurate when e_d is small. The reason is

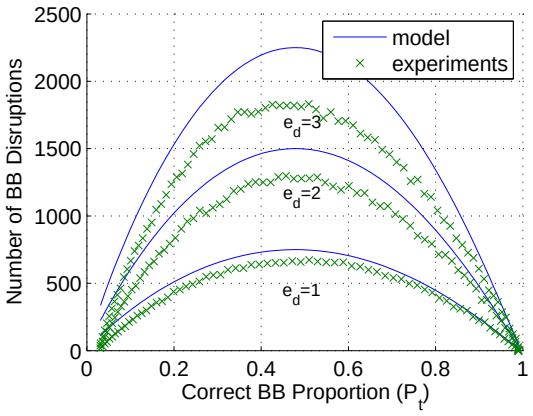


Figure 6.2: Numbers of BB disruptions for different numbers of detection-failure errors. The BB disruption is severe in the middle of the GA run, and mild when the GA is merely or nearly converged.

that Equation 6.4 is an overestimate that ignores the possibility of the recombination of two incorrect BBs producing a correct BB. When e_d is larger, the recombination of incorrect BBs has a higher probability to reproduce correct BBs. Note that the model indeed bounds the empirical data.

It is easily seen that BB disruption is severe when roughly half of the BBs in the population are correct; only few BB disruptions occur when the proportion of correct BBs is close to either 0 or 1. The observation suggests the following possible adaptive speedup scheme. Instead of recalculating the interaction model every generation, the interaction model is only updated every several generations at the beginning and the end of the GA run. Of course, it is non-trivial to estimate the degree of convergence of the GA for any given generation, and that leaves room for future work.

Finally, Equation 6.8 predicts that GAs hardly converge when $e_d > 2I\sqrt{m}$. In the experiments, since a BB is exchanged with a probability of 0.5, the number of BB disruptions is only half as modeled, $e_{critial} = 4I\sqrt{m}$. The critical number of errors versus the number of BBs is plotted in Figure 6.3. As shown in the figure, the model agrees with the results: $e_{critical}$ grows proportionally to the square root of the number of BBs, $\sqrt{m}$. Since the number of BB disruptions is overestimated, the estimation of $e_{critical}$ should be a underestimation.

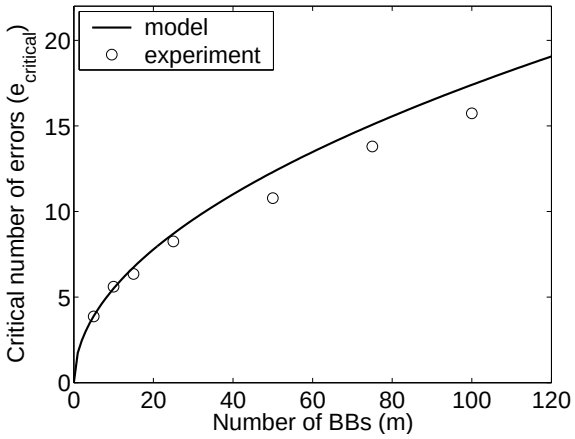


Figure 6.3: The critical number of errors versus the number of BBs. Roughly $e_{critical} = \Theta(\sqrt{m})$.

However, due to a finite population size, the empirical $e_{critical}$ is smaller than predicted.

6.2.4 Summary

To sum up, this section indicates that if the interaction model has e_d detection-failure errors, GA convergence time elongates by a factor of $\left(1 - \frac{e_d}{2I\sqrt{m}}\right)^{-1}$, where m is the number of BBs and I is the selection intensity. When e_d is greater than $2I\sqrt{m}$, GAs fail to find all correct BBs.

6.3 False Interaction and Convergence Time

The previous section discusses how detection failure affects GA convergence. This section investigates the other type of errors—false interaction.

6.3.1 Effective Exchange Length

The positions where false-interaction errors occur can be either fixed or random. Consider a problem with four BBs: BB_A, BB_B, BB_C, and BB_D. If the model builder always groups BB_A and BB_B together as BB_{AB}, the false-interaction error occurs at a fixed position. Since

two BBs of order k are grouped together, the population size required to ensure the presence of at least one copy of the correct BB_{AB} in the initial population is $\Theta(2^{2k})$. Note that whereas there is no false interaction, the population size dictated by BB supply is $\Theta(2^k)$. Even if mutation is introduced to generate the correct allele values in BB_{AB}, the number of mutations needed is still $\Theta(2^{2k})$. False-interaction errors occur at fixed position when the interaction-detection method is applied off-line, or when the interaction model is biased to some particular positions.

On the other hand, if the interaction-detection method is applied every generation and is unbiased, false-interaction errors occur at random positions. In other words, the interaction model randomly groups BBs by mistake due to sampling noises coming from a finite population size. This section focuses on those false-interaction errors that occur at random positions. In this case, the population size dictated by BB supply is $\Theta(2^k)$, the same as that when no false interaction occurs. If the interaction model mistakenly groups two BBs together in the current generation, with a non-zero probability, the interaction model will not group these two BBs together again in the next generation, and hence the crossover operator will mix them. As one can expect, the mixing rate becomes somewhat lower. The remainder of this section focuses on how false interaction affects BB mixing.

Consider the following two scenarios: (1) the interaction model groups BB_1 and BB_2 together, and (2) the interaction model identifies BB_1 and BB_2 correctly as two separate BBs, and the BB-wise uniform crossover by chance transfers BB_1 and BB_2 together. These two scenarios produce the same crossover result, but occur with different probabilities. In the first scenario, the information of BB_1 and BB_2 are transferred together with a probability of 1 while the second scenario does the same thing with a probability of 0.5. The difference changes the *effective exchange length* (EEL), defined as the effective number of BBs exchanged during crossover. Note that for a problem with m BBs, the minimal EEL is 0 and the maximal EEL is $\frac{m}{2}$ because exchanging m' BBs is the same as exchanging $(m-m')$ BBs.

It is not difficult to calculate EEL for uniform crossover with perfect BB information. Suppose that the problem has m BBs. The probability that the crossover exchanges 0 BB can be calculated as $\frac{C_0^m + C_m^m}{2^m}$, where C_b^a is the binomial coefficient. A similar calculation can be done for every different exchange length, and subsequently EEL can be expressed as follows.

$$EEL_{unif} = 2^{-m}\left[\frac{m}{2}\cdot C_{\frac{m}{2}}^m + \Sigma_{i=0}^{\frac{m}{2}-1} i\cdot\left(C_i^m + C_{m-i}^m\right)\right], \tag{6.9}$$

where m is assumed to be even for simplicity. Equation 6.9 can be simplified by the following three arithmetic relations: (1) $C_b^a = C_{a-b}^a$, (2) $b\cdot C_b^a = a\cdot C_{b-1}^{a-1}$, for $a, b \geq 1$, and (3) $\Sigma_{i=0}^a C_i^a = 2^i$.

$$\begin{aligned}
EEL_{unif} = &\quad 2^{-m}\left[\left(2\Sigma_{i=1}^{\frac{m}{2}} i\cdot C_i^m\right) - \tfrac{m}{2}\cdot C_{\frac{m}{2}}^m\right] \\
= &\quad 2^{-m}\left[\left(2\Sigma_{i=1}^{\frac{m}{2}} m\cdot C_{i-1}^{m-1}\right) - \tfrac{m}{2}\cdot C_{\frac{m}{2}}^m\right] \\
= &\quad 2^{-m}\left[\left(m\Sigma_{i=0}^{m-1} C_i^{m-1}\right) - \tfrac{m}{2}\cdot C_{\frac{m}{2}}^m\right]. \\
= &\quad \frac{m2^{m-1} - \frac{m}{2}C_{\frac{m}{2}}^m}{2^m} \\
= &\quad \frac{m}{2}\left(1 - \frac{C_{\frac{m}{2}}^m}{2^m}\right).
\end{aligned} \tag{6.10}$$

Consider the Stirling approximation (Stirling, 1730), $m! \simeq \sqrt{2\pi m}\cdot m^m e^{-m}$. $C_{\frac{m}{2}}^m$ can be approximated as $2^m\sqrt{\frac{2}{\pi m}}$. Therefore, EEL of uniform crossover can be approximated as

$$EEL_{unif} \simeq \frac{m}{2}\left(1 - \sqrt{\frac{2}{\pi m}}\right). \tag{6.11}$$

Unsurprisingly, for a large m, uniform crossover on average exchanges $\frac{m}{2}$ BBs, which is the maximal information exchange.

Consider a problem with 4 BBs. If the interaction model is perfect, the probability that crossover operator exchanges $\{0, 1, 2, 3, 4\}$ BBs is $\frac{1}{2^4}\{C_0^4, C_1^4, C_2^4, C_3^4, C_4^4\} = \frac{1}{16}\{1, 4, 6, 4, 1\}$, respectively. The corresponding EEL is $0\cdot\frac{2}{16} + 1\cdot\frac{8}{16} + 2\cdot\frac{6}{16} = 1.25$. Suppose that the interaction

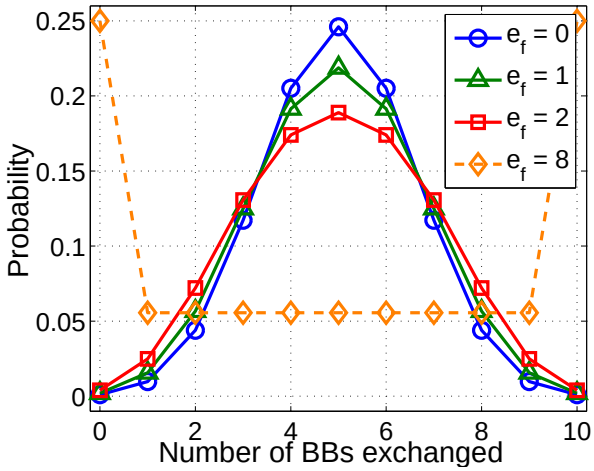


Figure 6.4: Probabilities of a certain number of exchanged BBs for a problem with 10 BBs. Parameter e is the number of false-interaction errors in the interaction model. EEL for $e_f = 0$ is roughly 10/2=5, and that for $e_f = 8$ is roughly 10/8=1.25.

model mistakenly groups BB_1 and BB_2. If BB_1 and BB_2 are not exchanged, the probability that $\{0, 1, 2, 3, 4\}$ BBs are exchanged is $\frac{1}{4}\{1, 2, 1, 0, 0\}$; if BB_1 and BB_2 are exchanged, the probability that $\{0, 1, 2, 3, 4\}$ BBs are exchanged is $\frac{1}{4}\{0, 0, 1, 2, 1\}$. The above two cases have equal probability to occur. Therefore, the probability that $\{0, 1, 2, 3, 4\}$ BBs are exchanged given 1 false-interaction error is $\frac{1}{8}\{1, 2, 2, 2, 1\}$, respectively. The corresponding EEL is reduced from 1.25 to 1.00 when the interaction model has 1 false-interaction error.

When considering 2 false-interaction errors, the calculation is similar but becomes somewhat more complicated because the interaction model may contain two chunks of two BBs or one big chunk of three BBs. The effect of false interaction on the number of BBs exchanged for a problem with 10 BBs is plotted in Figure 6.4. The dashed line represents $e = 8$. Note that the maximal number of false-interaction errors that still yields information exchange is $(m - 2)$. The m BBs become only one chunk when there are $(m - 1)$ false-interaction errors, and hence no information exchange is possible during crossover. With $(m - 2)$ false-interaction errors, the interaction model has only two big chunks of BBs. With a probability of 0.5, exactly one chunk is exchanged, and the effect is similar to two-point crossover. The corresponding EEL is roughly $\frac{1}{2} \cdot \frac{m}{4} = \frac{m}{8}$, where $\frac{m}{4}$ is the EEL of two-point crossover.

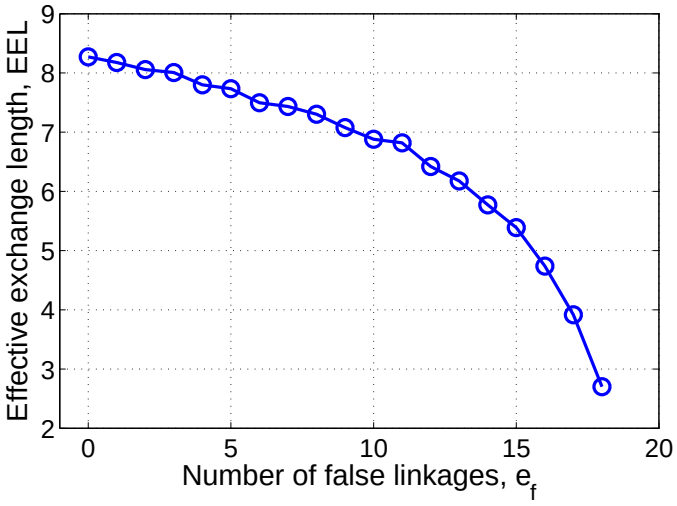


Figure 6.5: Relationship between EEL and the number of false-interaction errors, e_f. The problem is composed of 20 BBs.

The relationship between EEL and false interaction for a problem with 20 BBs is shown in Figure 6.5. The EEL roughly drops from $\frac{m}{2} = 10$ for no false interaction, $e_f = 0$, to $\frac{m}{8} = 2.5$ for the maximal number of false-interaction errors, $e_f = (m - 2) = 18$.

In summary, a perfect interaction model with uniform crossover has an EEL of $\frac{m}{2}$; and a nearly worst interaction model that contains $(m - 2)$ false-interaction errors has an EEL of $\frac{m}{8}$. The worst interaction model with $(m - 1)$ false-interaction errors has only one big chunk, and the GA does not work unless with an exponentially large population.

6.3.2 Effective Exchange Length and Convergence Time

This subsection investigates the relationship between EEL and GA convergence time. As shown before, uniform crossover with an exchange probability of 0.5 on average exchanges $\frac{m}{2}$ BBs. This statement can be generalized for other exchange probabilities by examining the nature of binomial distribution. The mean of a binomial distribution is Np, where N is the number of Bernoulli trials and p is the probability that the Bernoulli trial gives a true value. Although the calculation is somewhat different, it is not difficult to show that for large m, the EEL of uniform crossover with exchange probability p is approximately mp

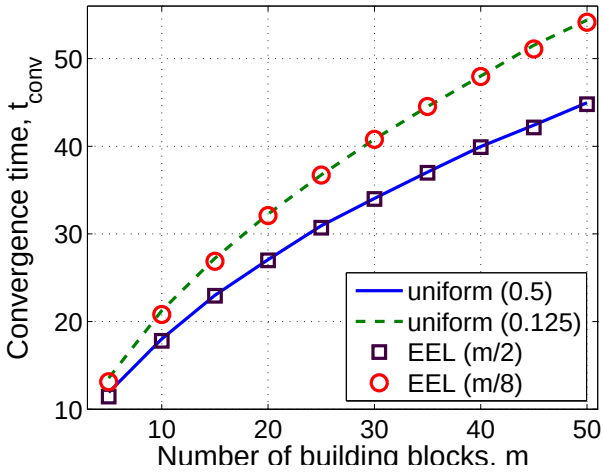


Figure 6.6: Convergence time for different crossover operators. EEL-crossover with crossover length η has the roughly the same convergence time as uniform crossover with exchange probability η/m.

using a similar derivations in the previous subsection. Therefore, the effect of the crossover with EEL η should be similar to uniform crossover with exchange probability $\frac{\eta}{m}$. Design a crossover operator, *EEL-crossover*, which randomly chooses η BBs and then exchanges them. Figure 6.6 illustrates that EEL-crossover with crossover length η is similar to uniform crossover with exchange probability η/m.

Recall that the convergence time can be modeled as (Mühlenbein & Schlierkamp-Voosen, 1993; Thierens & Goldberg, 1994):

$$t_{conv} = \frac{c_c\sqrt{\ell}}{I}, \tag{6.12}$$

where ℓ is problem size, I is selection intensity, and c_c is a constant. Assuming the order of BBs is fixed, the problem size, ℓ, is proportional to the number of BBs, m. In the derivations of Equation 6.12 (Chapter 1.4), binomial distribution is assumed. In other words, if the current population has a proportion p of correct BBs, the mean and variance of the number of correct BBs for each individual are mp and $\sqrt{mp(1-p)}$, respectively. Rabani, Rabinovich, and Sinclair (1998) derived bounds for the relaxation time that uniform crossover needs to

randomize the population as follows.

$$\frac{\ln n}{2\ln q^{-1}} \leq \tau_{unif} \leq \frac{2\ln n}{\ln q^{-1}}, \tag{6.13}$$

where q is the probability that two positions are not separated. In the case of uniform crossover with exchange probability p, $q = p^2 + (1-p)^2$. For the exchange probability reduced from $\frac{1}{2}$ to $\frac{1}{8}$, the relaxation time increases roughly by a magnitude of 2.8. Therefore, the distribution of correct BBs can no longer be modeled as a binomial distribution. The variance of the number of correct BBs should be less than $\sqrt{mp(1-p)}$, making the GA converges more slowly. Followed by the notion of facetwise modeling in Goldberg (2002) and empirical findings (Figure 6.6), Equation 6.6 still yields a good approximation, but with a different constant $c_c' = r_f \cdot c_c$ when uniform crossover has an exchange probability smaller than 0.5. The parameter r_f is referred as the *elongation factor* for false interaction. The experiment conducted in Figure 6.6 indicates that for m varying from 10 to 50, the elongation factor for $(m-2)$ false-interaction errors only varies roughly from 1.18 to 1.21. The elongation factor is not tightly related to the problem size, and hence r_f is treated as invariant to m.

The previous subsection indicates that an inaccurate interaction model with false interaction reduces the corresponding EEL. The crossover with a reduced EEL η due to an imperfect model has a similar effect as the uniform crossover with a smaller exchange probability $p = \frac{\eta}{m}$. The reduction of the exchange probability elongates the convergence time $t_{conv} = \frac{r_f \cdot c_c \sqrt{m}}{I}$. Figure 6.7 shows the relationship between e_f and t_{conv}. It is easily seen that η is a function of e_f, and r_f is a function of $p \simeq \frac{\eta}{m}$. Both relationships are non-linear and difficult to model quantitatively. However, the qualitative model provides a way to compare the impacts of detection failure versus false interaction.

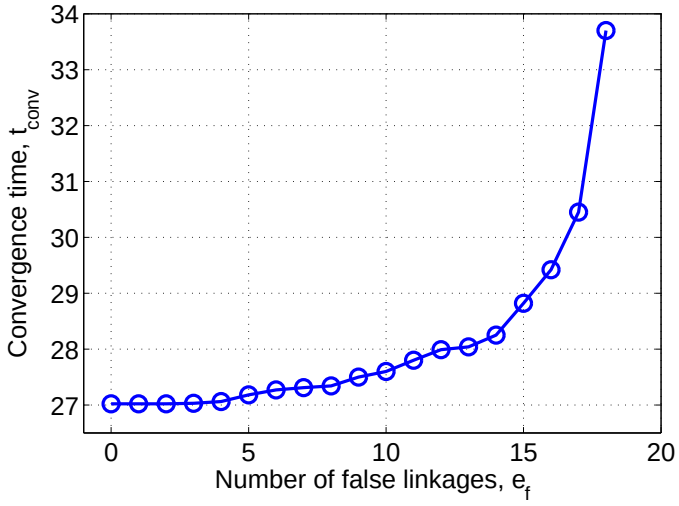


Figure 6.7: The relationship between the convergence time and the number of false-interaction errors. The testing problem contains 20 BBs.

6.4 Detection Failure vs. False Interaction

This section compares the effects of detection failure and false interaction. Recall that $t_{conv}(e_d) = \frac{1}{1-\frac{e_d}{2I\sqrt{m}}}\frac{\pi\sqrt{m}}{I}$ for e_d detection-failure errors, and hence the corresponding elongation factor r_d is $\frac{1}{1-\frac{e_d}{2I\sqrt{m}}}$. The quality of an interaction model can be defined as the probability that the model produces either a detection-failure error or a false-interaction error. In term of the impact on GA convergence time, detection failure outweighs false interaction when the following condition is satisfied.

$$\begin{aligned} & r_f < r_d \\ \Rightarrow \quad & r_f < \frac{1}{1-\frac{e_d}{2I\sqrt{m}}} \\ \Rightarrow \quad & m < \frac{1}{\left[2I\left(1-\frac{1}{r_f}\right)\right]^2}{e_d}^2. \end{aligned} \tag{6.14}$$

Recall that empirically r_f varies only from 1 to 1.2 for e_f from 0 to $(m-2)$. The term $\frac{1}{\left[2I\left(1-\frac{1}{r_f}\right)\right]^2}$ can be treated as a constant c_s over a wide range of m. In addition, assume that both types of errors occur with an equal probability p. The quality Q of the interaction model can then be defined as $(1-p)$. Given that the number of detection-failure errors ranges

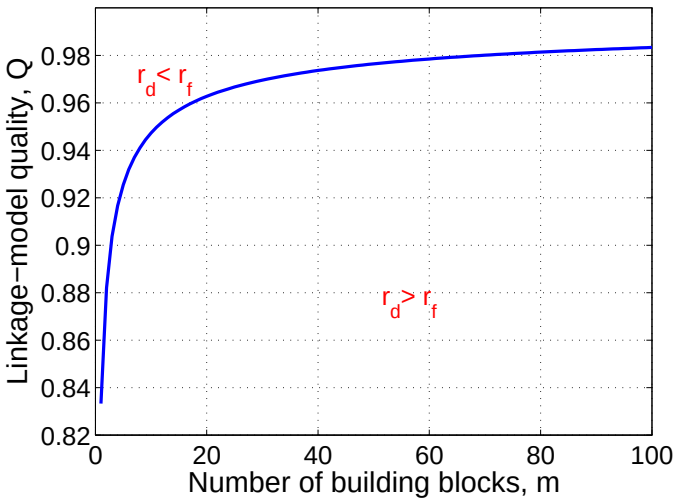


Figure 6.8: The control map of the two types of errors, where r_d is the elongation factor of detection failure, and r_f is that of false interaction. In most problems, detection failure affects the convergence time more severe than false interaction. False interaction dominates only when the problem size is small and the interaction model is very accurate.

from 0 to m, Q can be approximated as $1 - p \simeq 1 - \frac{e_d}{m}$. With arithmetic manipulations, Inequality 6.14 can be re-expressed as

$$m > \frac{1}{c_s(1-Q)^2}. \tag{6.15}$$

Given the empirical findings that selection intensity $I \simeq 0.5$ for problems composed of trap$_5$ functions, c_s is roughly 36. Figure 6.8 plots a control map of these two types of errors. In most problems, detection failure affects the convergence time more severely than false interaction. False interaction dominates only when the problem size is extremely small or when the interaction model is extremely accurate.

Two important insights can be gained from those models. One is that both types of errors result in a longer convergence time; for a fixed number of errors, the elongation factor for detection failure relates inversely to the problem size while that for false interaction is virtually independent of the problem size. The other is that for most problems, the interaction-detection algorithm should emphasize more on eliminating detection failure than

false interaction.

6.5 Overall Computational Time

This section models the overall computational time using the time-to-convergence models in previous sections. The overall computational time is then used to derive an optimal decision for which interaction model should be used. A similar method of modeling can be found in Sastry (2002).

Assuming that the GA operators consumes α computational time each generation, if an interaction model contains e errors and consumes β computational time, the overall computation time is then

$$T = t_{conv}(e) \cdot (\alpha + \beta). \tag{6.16}$$

Since the impact of detection failure on GA convergence is stronger than that of false interaction, e is assumed to be the number of detection-failure errors. Consider two interaction-detection methods, M_1 and M_2, which misidentifies e_1 and e_2 BBs and consumes α_1 and α_2 computational time, respectively. The ratio of the overall computational time of GAs that adopt those two interaction-detection methods is given by

$$\frac{T_{M_1}}{T_{M_2}} = \frac{2I\sqrt{m} - e_2}{2I\sqrt{m} - e_1} \cdot \frac{\alpha + \beta_1}{\alpha + \beta_2}. \tag{6.17}$$

If the ratio is less than 1, method M_1 should be used, and vice versa.

The above equation is difficult to use in practice mainly because the number of BBs that an interaction model misidentifies is difficult to estimate. Nevertheless, the equation gives some insights and mathematical foundation to the following observations.

1. When the fitness function evaluation is computationally expensive ($\alpha \gg \beta$), the first term ($\frac{2I\sqrt{m}-e_2}{2I\sqrt{m}-e_1}$) dominates the decision, and a time-consuming but more accurate interaction detector is favored.

2. On the contrary, when the fitness function evaluation is relatively computationally inexpensive ($\alpha \ll \beta$), the second term ($\frac{\alpha+\beta_1}{\alpha+\beta_2}$) dominates the decision, and a less accurate but computational efficient interaction detector is favored.

When errors are few ($e_1, e_2 \ll \sqrt{m}$) and the computational cost for the interaction model builder is relatively cheap compared to the GA operators ($\beta_1, \beta_2 \ll \alpha$), the above equation can be approximately simplified as

$$\frac{T_{M_1}}{T_{M_2}} = 1 + \frac{e_1 - e_2}{2I\sqrt{m}} + \frac{\beta_1 - \beta_2}{\alpha}. \tag{6.18}$$

The above equation suggests the following definitions: The *quality*[1] of an interaction model is $Q = 1 - \frac{e}{2I\sqrt{m}}$ and the *relative cost* for an interaction model is $c = \frac{\beta}{\alpha}$. For any interaction model with $Q < 0$ or $e > 2I\sqrt{m}$, the GA is difficult to converge. Suppose that two interaction-detection methods M_1 and M_2 have qualities Q_1 and Q_2 and relative costs c_1 and c_2, respectively. By defining $\triangle Q = Q_1 - Q_2$ and $\triangle c = c_1 - c_2$, the decision ratio becomes

$$\frac{T_{M_1}}{T_{M_2}} = 1 + (\triangle c - \triangle Q). \tag{6.19}$$

Therefore, if $\triangle c < \triangle Q$, M_1 is better; otherwise, M_2 is better. Consider M_2 as an evaluation relaxation version of M_1: M_2 is more computational efficient but less accurate than M_1 ($\triangle c > 0$ and $\triangle Q > 0$). The evaluation relaxation is worthwhile when $\triangle c > \triangle Q$, or in other words, when the savings in relative cost is greater than the loss of quality.

[1]Note that the quality defined here ($Q = 1 - \frac{e}{2I\sqrt{m}}$) is different from before ($Q = 1 - \frac{e}{m}$) for different purposes.

6.6 Summary and Conclusions

This chapter investigates the two types of errors that may occur in an interaction model—detection failure and false interaction. The relationship between these errors and GA convergence time is modeled and empirically verified. From those facetwise models, one can recognize three keys to achieve efficient and effective recombination:

Minimization of BB disruptions. GA convergence time is elongated approximately by a factor of $\frac{1}{1-\frac{e}{2I\sqrt{m}}}$, given the number of BB disruptions is e. GAs hardly converge when the number of BB disruptions is greater than $\Theta(\sqrt{m})$ except with an exponentially large population.

Maximization of effective exchange length. GAs converge faster when the recombination operator has a larger EEL. Intuitively, having the amount of information exchange maximized during the recombination is preferred.

Nondeterminism of information exchange. The information exchange needs to be nondeterministic. Every pair of BBs should have a non-zero probability to be mixed. Otherwise, the problem difficulty would increase. If two BBs never have a chance to get mixed, the population size required to solve that subproblem increases from $\Theta(2^k)$ to $\Theta(2^{2k})$, where k is the order of BBs.

The next chapter investigates problems with overlap and demonstrates how to achieve efficient and effective recombination based on these principles.

Chapter 7

Finding Extrema for Problems with Overlap: DSMGA++

The previous chapter investigates how the quality of the interaction model affects the convergence of GAs. It recognizes three keys to achieve effective and efficient recombination:

1. Minimization of disruption.
2. Maximization of effective exchange length.
3. Nondeterministic information exchange.

This chapter investigates problems with overlap and tempts to develop recombination methods to conquer the overlap difficulty. Starting from a simple problem with cyclically overlapping BBs, this chapter proposes a recombination method, called `minimalcut`, based on the above principles. However, when the underlying topology of overlapping becomes more complicated, `minimalcut` fails. This chapter then proposes several essential modifications by recognizing the reasons for `minimalcut` failure. In particular, a problem with much more overlaps—2D spin glasses—is investigated in this chapter. This chapter concludes with three keys to conquer the overlap difficulty:

1. Preservation of alternative solutions.
2. Proper sequencing.
3. Well-informed decision.

The reasons behind these ideas will be explained in detail in this chapter. Since the method developed in this chapter is based on DSMGA and it handles one more type of

interactions, it is named as DSMGA++. The experiments in this chapter also demonstrate the off-line usage of the DSM analysis, and the pros and cons for performing model building off-line are discussed.

7.1 `MinimalCut` on a Problem with Cyclically Overlapping BBs

Given the argument that detection-failure errors elongate the convergence time much more than false-interaction errors, a simple recombination strategy can be proposed as follows: Treat the whole problem as two big BB chunks. As indicated before, increasing the mixing rate by raising the number of cross sites results in more BB disruptions. Therefore, the recombination strategy when dealing with a group of overlapping BBs is as follows:

Algorithm 1 The `minimalcut` recombination strategy.

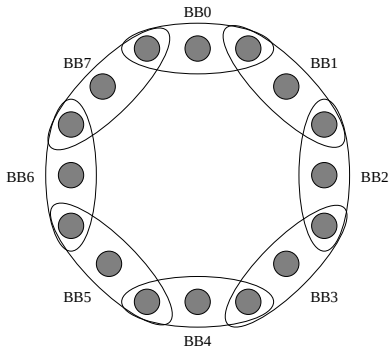
1. Perform BB identifying algorithm to capture the overlapping topology.

2. Construct a graph $G = (V, E)$ where the nodes are BBs, and the edges are overlapping relations between BBs. There is an edge between two BBs if and only if the two BBs overlap.

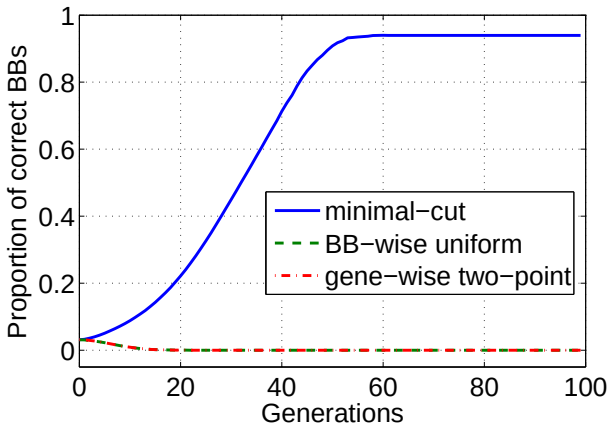
3. Randomly choose two nodes n_1 and n_2. Then partition the graph G into two subgraphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ that satisfy the conditions: $n_1 \in V_1$, $n_2 \in V_2$, and $|E| - |E_1| - |E_2|$ is minimal.

In other words, the two chunks need to be chosen at random, and the choice disrupts minimal number of overlapping BBs.

The `minimalcut` algorithm is demonstrated on a simple problem with cyclically overlapping BBs (Figure 7.1(a)). The problem has an overlapping length of 2. For example, a problem with 3 BBs has the fitness defined as $trap_5(y_1y_2y_3y_4y_5) + trap_5(y_4y_5y_6y_7y_8) + trap_5(y_7y_8y_9y_1y_2)$, where $\{y_i\}$ is a random permutation of genes and $trap_5$ is a trap function



(a) A problem with 8 overlapping BBs.



(b) The convergence time for the cyclically overlapping problem.

Figure 7.1: The GA convergence on a problem with cyclically overlapping BBs. The fitness of each BB is calculated by a trap_5 function. The overlapping length is 2. Exchanging only BB_0 disrupts both BB_1 and BB_7 for the problem with 8 BBs. Three different crossover operators are used. Both allele-wise two-point crossover and BB-wise uniform crossover fail, and the lines are overlapped. Only `minimalcut`, which respects both BB structure and overlapping topology, succeeds in finding high-quality solutions.

of order 5. Note that the arrangement of genes is randomly shuffled so the problem in general is not of tight modularity.

Since the BB structure is a circle, the minimal disruption occurs when the circle is cut at only two points. To maximize EEL, once a cutting point is chosen, the other cutting point should be chosen at exactly the opposite position. Finally, to achieve nondeterministic information exchange, the first cutting point should be chosen at random. Three different crossover operators are tested: (1) allele-wise two-point crossover, (2) BB-wise uniform crossover, and (3) `minimalcut`. The allele-wise two-point crossover does not exploit the information of BBs. The BB-wise uniform crossover is similar to an ordinary allele-wise uniform crossover, except the operands are BBs instead of alleles. The result is shown in Figure 7.1(b).

To examine the scalability, `minimalcut` is applied to the problem with different number of cyclically overlapping BBs. The result is shown in Figure 7.2. The solid line represents

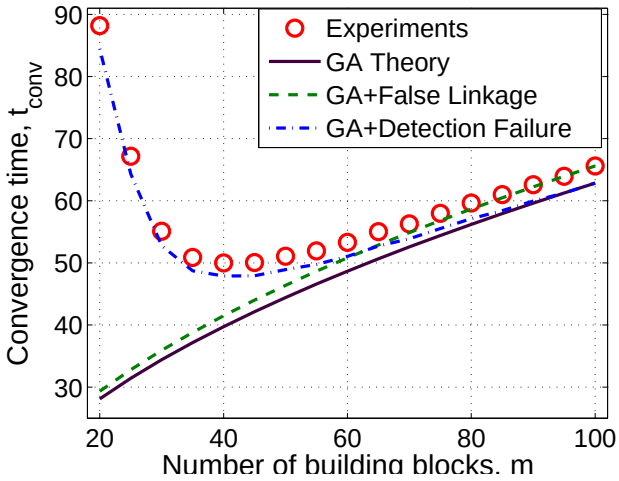


Figure 7.2: The scalability of `minimalcut` on the cyclically overlapping problem with different problem sizes. The solid line represents the GA convergence theory. The dashed line considers only the effect of false interaction, and the dotted line considers only the effect of detection failure.

the GA convergence theory. The dashed line considers only the effect of false interaction, and the dotted line considers only the effect of detection failure. The figure indicates a much longer convergence time when the problem size is small. The reason is that `minimalcut` always disrupts two BBs. Practically, when the problem size is very small, it is beneficial to simply treat the overlapping BBs as one big BB.

Note that although the behavior of `minimalcut` on the problem with cyclically overlapping BBs seems similar to an ordinary two-point crossover, there is a significant difference. The ordinary two-point crossover does not respect the interaction topology. It will succeed only for problems with tight modularity. For problems with overlapping BBs and random modularity, (1) if no BB-identification method is applied, the GA fails, and (2) if BBs are correctly identified but a crossover that does not respect the overlapping topology is adopted, the GA fails, too. When dealing with problems with overlap, crossover operators that respect both modularity and overlap topology are essential for GA success. Note that the strategy of splitting the group of overlapping BBs into two chunks should only be applied to overlapping BBs. For a problem containing both overlapping and non-overlapping BBs, BB-

wise uniform crossover should be applied to those non-overlapping BBs to achieve maximal mixing.

7.2 A Problem with More Overlap: 2D Ising Spin Glasses

The `minimalcut` algorithm in the previous section scales well on the problem with cyclically overlapping BBs. However, it is not expected to be the solution to every problem with overlap since the degree of overlapping in the example is only moderate, where a gene belongs to at most two BBs, and a BB overlaps with at most two other BBs. The following sections focus on a real-world problem where the underlying topology of overlapping is more complicated—the Ising spin glass problem.

The study of spin glasses (Fischer & Hertz, 1991) has attracted considerable attention in statistical physics and condensed matter physics. Researchers have been studying the optimization of the Ising spin-glass problem by using GAs because of interesting properties like the symmetry, large number of local optima, scalability of the problem size, and overlapping BBs (Pelikan, Ocenasek, Trebst, Troyer, & Alet, 2004; Pelikan & Goldberg, 2003; Pelikan & Mühlenbein, 1999; Naudts & Naudts, 1998; van Hoyweghen, Goldberg, & Naudts, 2002; van Hoyweghen, 2001; Mühlenbein, Mahnig, & Rodriguez, 1999).

The physical state of an Ising spin-glass system is defined by (1) a set of spins $\{\sigma_0, \sigma_1, \cdots, \sigma_{l-1}\}$, where σ's are Ising variables with values -1 or +1, and (2) a set of coupling constants J_{ij} relating spins σ_i and σ_j. The simplest spin-glass Hamiltonian is defined as:

$$H(\sigma) = -\Sigma_{i,j=0}^{n-1} J_{ij}\sigma_i\sigma_j. \tag{7.1}$$

The objective is to find the ground state of the spin-glass system. In other words, given a set of coupling constants J_{ij}, the task is to find a set of spin values $\{\sigma_i\}$ that maximizes

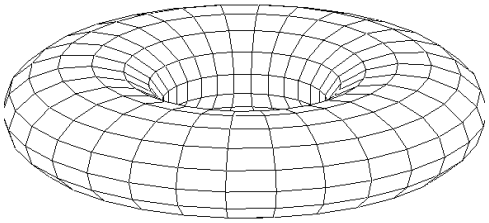


Figure 7.3: A torus. It is the topology of the 2D spin-glass problems with periodic boundary condition.

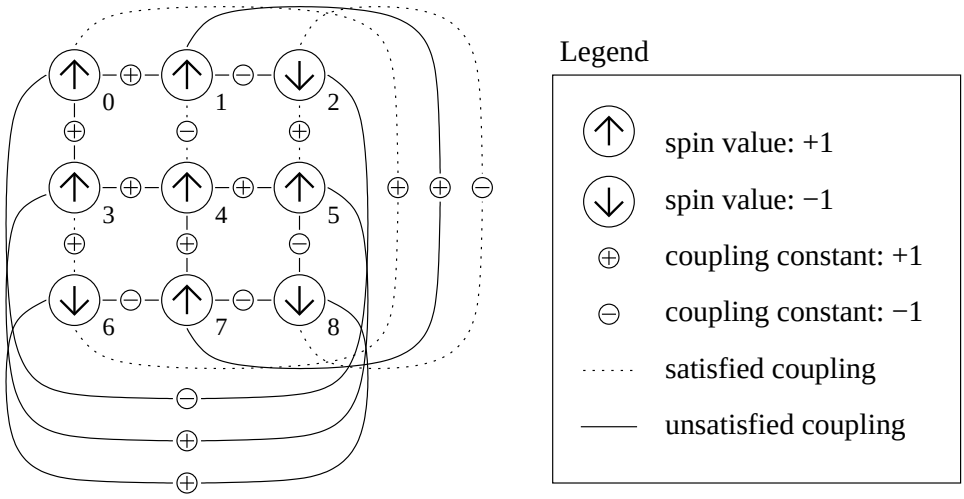


Figure 7.4: An example of a 3×3 spin-glass problem. There are total 13 couplings are satisfied and 5 couplings are unsatisfied, which gives the fitness of 8.

Equation 7.1.

Here the coupling constants are restricted to binary, $J_{ij} \in \{-1, +1\}$. Also, this chapter only considers the two dimensional case where the spins are arranged in a 2D lattice and every spin couples with its four neighbors. To approximate the behavior of a large-scale system, the periodic boundary condition is adopted, making the 2D lattice a torus (Figure 7.3). Figure 7.4 shows an example of a three by three spin-glass system. There are 13 satisfied couplings and 5 unsatisfied couplings, which yields a fitness value of 8.

For the Ising spin-glass problem in its general form, to verify if a set of spin values is the ground state is known to be NP-complete.[1] However, in the 2D special case, several polynomial algorithms exist, and the current best one scales as $\Theta(l^{3.5})$, where l is the problem

[1]The Ising spin-glass problem is equivalent to the `MIN-CUT` problem, which is NP-complete (Monien & Sudborough, 1988), by polynomial reduction.

size (Galluccio & Loebi, 1999a; Galluccio & Loebi, 1999b).

7.2.1 The BB structure of the 2D spin-glass problem

In this chapter, interaction detection is performed off-line. The interaction model is obtained from arithmetic manipulation and off-line DSM analysis. Two reasons lead to perform interaction detection off-line: (1) to focus on the recombination strategy and ignore the factor of modularity identification error, and (2) to demonstrate how to utilize DSM analysis off-line on smaller-scale problems and apply the obtained knowledge to larger-scale problems.

The first two terms of the fitness function of the spin-glass instance shown in Figure 7.4 are:

$$H(\sigma) = \sigma_0\sigma_1 - \sigma_1\sigma_2 + \cdots . \tag{7.2}$$

If $(\sigma_0, \sigma_1) = (+1, +1)$ or $(-1, -1)$, the fitness will increase by one from the first term. Similarly, $(\sigma_1, \sigma_2) = (-1, +1)$ or $(+1, -1)$ contributes one to the fitness. Recall that BBs are groups of minimal, sequential, and superior genes. Thus, σ_0 and σ_1 form a BB while σ_1 and σ_2 form another one. The BB structure of the 2D spin-glass problem is shown in Figure 7.5. Therefore, a 2D spin-glass problem of size l contains $2l$ BBs; every gene belongs to four different BBs, and every BB overlaps with six other BBs.

The BB structure can also be obtained by DSM analysis. Given a 2D spin-glass problem with randomly assigned coupling constants, generate a population of chromosomes with randomly initialized spin-values. Evaluate every chromosome using the fitness function, perform selection, and then create a DSM over those selected chromosomes. The average neighborhood information can be retrieved by averaging the DSMs over many spin-glass instances. Finally, applying the DSM clustering technique yields the same BB structure.

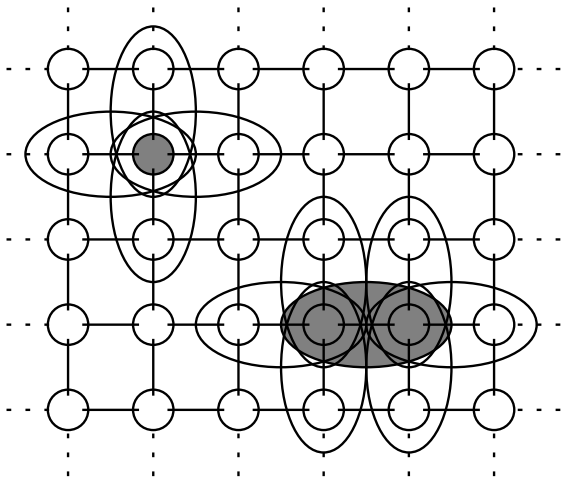


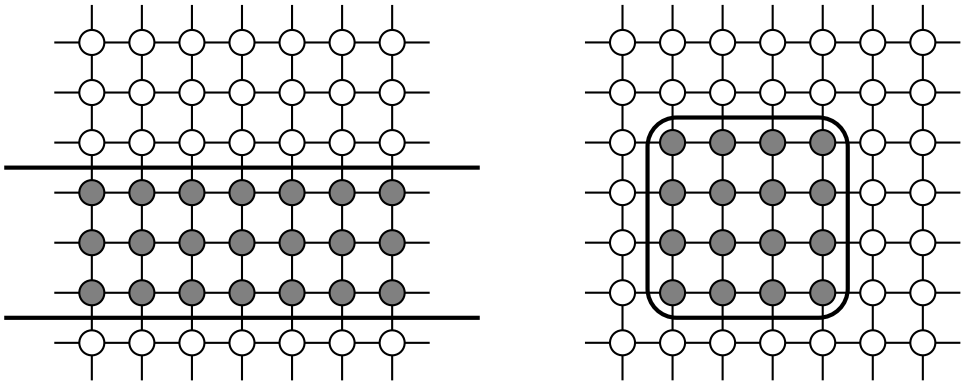
Figure 7.5: The BB structure of the 2D spin-glass problem. Every pair of coupled genes forms a BB. Therefore, in a 2D spin-glass problem of size l, there are totally $2l$ BBs. The blackened gene shows that every genes belongs to four different BBs, and the blackened BB shows that every BB overlaps with other six BBs.

7.3 Trial One: `MinimalCut`

As a first attempt, `minimalcut` is applied to the 2D spin-glass problem. As stated before, finding the minimal cut of a graph is NP-hard in general. However, for the 2D spin-glass problem, finding a minimal cut is not difficult.

Two possible ways of minimally cutting the overlapping graph of the 2D spin-glass problem are shown in Figure 7.6. The first method is similar to the two-point crossover of a simple GA when spins are arranged by a specific order (Figure 7.3). In this method, the number of BB disruptions is always $2\sqrt{l}$, and the effective exchange length (EEL) can be as large as $\frac{l}{2}$. In the second method, the number of BB disruptions is $4r$, and EEL is r^2. From Chapter 6.2, the number of BB disruptions needs to be less than $2I\sqrt{m}$, where I is selection intensity and $m = 2l$ in this case, for a successful GA convergence. By constraining the number of BB disruptions to be $\sqrt{l}$, calculation shows that $r = \frac{\sqrt{l}}{4}$ and EEL=$\frac{l}{16}$.

Figure 7.8 shows the empirical scalability test of a simple GA with these two cutting methods. The data points are medians of the results for 50 independent 2D spin-glass problem instances with randomly initialized coupling constants. For each problem, 10 bisection



(a) `minimalcut1` (b) `minimalcut2`

Figure 7.6: Two possible cutting ways for the `minimalcut` method. The information carried by those gray genes is mixed with that carried by the others. Given a problem with l spins, `minimalcut1` causes $2\sqrt{l}$ disruptions; if the square in Figure 7.6(b) is r by r, `minimalcut2` causes $4r$ disruptions.

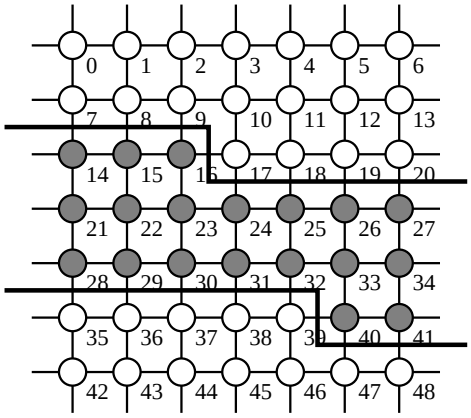
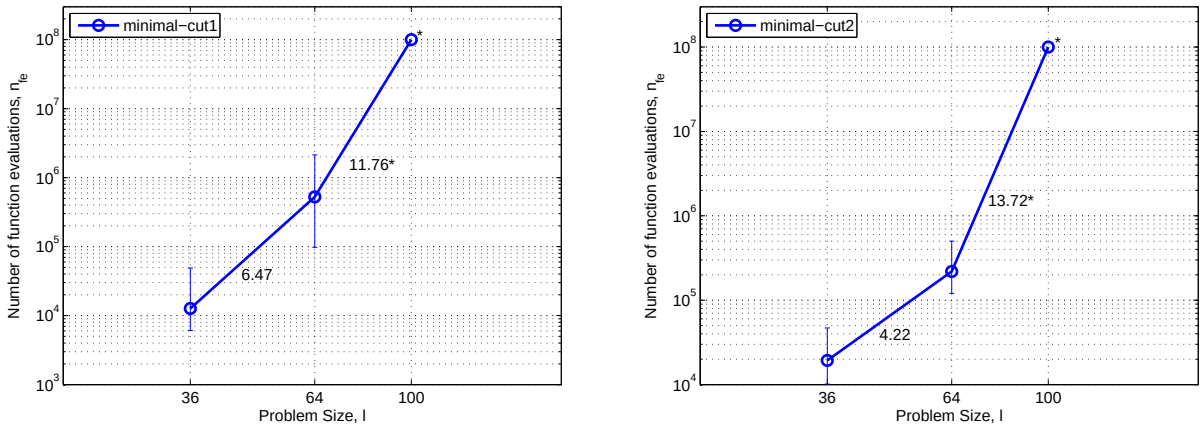


Figure 7.7: The information exchange and disruption of `minimalcut` is similar to that of two-point crossover when genes are arranged in the specific order.

runs are performed to find the minimal population size for 10 consecutively successes of finding the global optima. Using the averaged minimal population size yields the number of function evaluations that the GA needs for the 2D spin-glass problem. The lower error bar stands for the first quartile, and the upper error bar stands for the third quartile. Basically, none of these two cutting methods scales well on the 2D spin-glass problem. For most of the problems with 10 by 10 spins, the GA does not converge even after 10^8 function evaluations.



(a) minimalcut1. (b) minimalcut2.

Figure 7.8: Scalability of `minimal-cut` on the 2D spin-glass problem. The numbers on the figure indicate the slopes. For the problem of 10 by 10 spins, more than 90% of the time, the GA does not find the optimal solutions within 10^8 number of function evaluations.

7.4 Trial Two: `MinimalCut` + Niching

Experiments in the previous section showed that `minimalcut` does not scale well on the 2D spin-glass problem. This happens due to several reasons, and one of them is lack of the ability to preserve alternative solutions.

Figure 7.9 shows a simple 2D spin-glass problem with six spins. The global optimal solution is that all spins are of the same direction, which yields six satisfied couplings and one unsatisfied coupling. Note that locally, spin number 1 (σ_1) and spin number 4 (σ_4) should be in the opposite direction. As indicated before, σ_1 and σ_4 form a BB. Therefore, the decision of the correct BB among all four possible combinations $(\sigma_1, \sigma_4) = (1,1), (1,-1), (-1,1), (-1,-1)$ cannot be make correctly in the early stage of the GA run. In other words, before the decision can be made correctly, the GA needs to preserve alternative solutions. The observation suggests adopting niching methods in the GA.

The empirical scalability result for `minimalcut` with restricted tournament replacement (RTR) shows a significant improvement (Figure 7.10). With RTR, the GA is able to find a global optimum for the problem with 16 by 16 spins within 10^6 function evaluations.

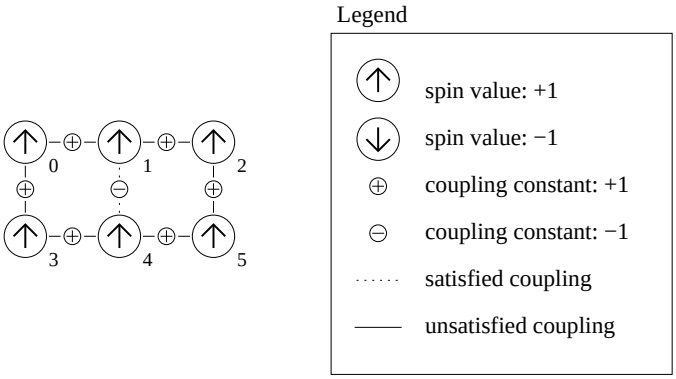
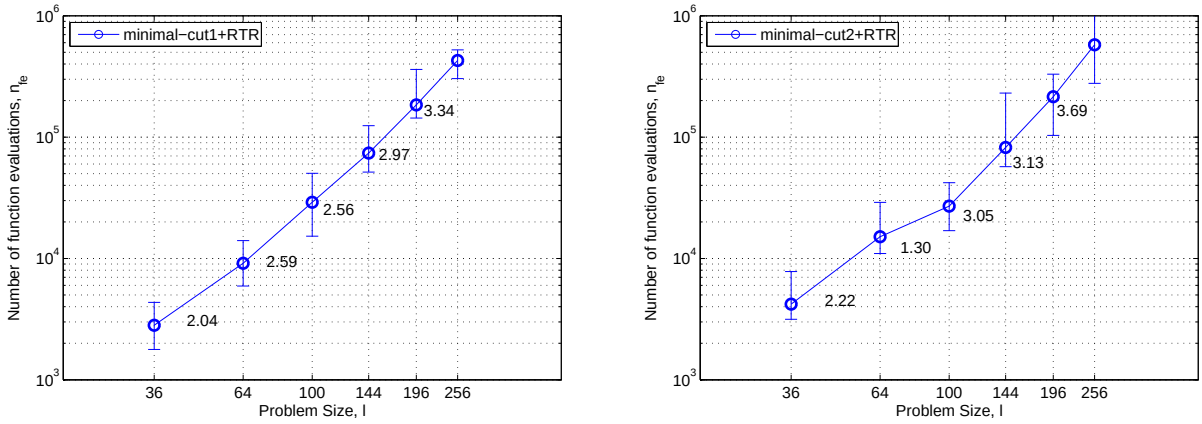


Figure 7.9: A 2D spin-glass problem to which the global optimal solution has one unsatisfied coupling.



(a) minimalcut1. (b) minimalcut2.

Figure 7.10: Scalability of `minimalcut+RTR` on the 2D spin-glass problem.

Nevertheless, in the log-log plot, the slopes of the data points increase rapidly with the problem size. According to the facetwise models developed in Chapter 6, the problem occurred for `minimalcut1` is due to too many BB disruptions, $\sqrt{2m}$ in total; the problem for `minimalcut2` is due to a too small EEL, $\frac{1}{16}$ of the total chromosome length, and moderate disruptions, $\sqrt{\frac{m}{2}}$ in total.

7.5 Trial Three: Sequencing + Niching

Possible improvements for the recombination fall into two aspects: (1) decrease the number of disruptions by considering the alleles of the overlapped genes, and (2) every BB has

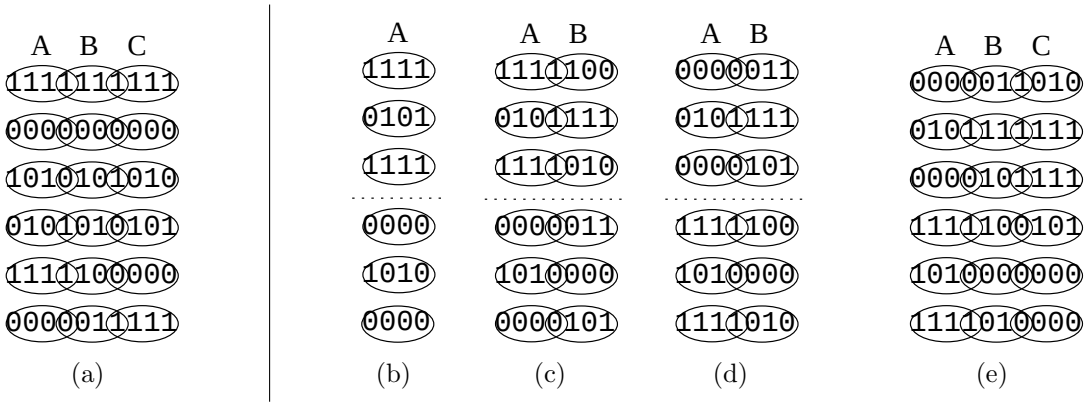


Figure 7.11: Non-disruptive population-wise crossover with non-cyclic overlaps.

different *strength*. If a certain number of BBs have to be disrupted, the weaker ones are preferred.

Suppose a problem with a population shown in Figure 7.11 has three overlapping BBs with no cycle. Crossover can be performed by considering the alleles of those overlapped genes to avoid disruptions. First partition the population by the alleles of the overlapped genes. In the example, BB_1 overlaps BB_2 with one gene, and therefore, partition the population into two sets where the overlapped gene of the chromosomes has a value of 0 in one set and a value of 1 in the other set. Now that the allele of the overlapped gene is coherent in both sets, population-wise shuffling can be performed between BB_1 and BB_2 within each set without disruption. Then the similar procedure is carried out to shuffle BB_2 and BB_3.

This modification should have a similar effect, if not better, as in the work of Tsuji, Munetomo, and Akama (2006), where they still adopt pair-wise crossover and treat two BBs as non-overlapping when the overlapped genes of the BBs have the same alleles. The crossover operator presented here examines the whole population to find those chromosomes with overlapped genes having the same alleles, and then performs a population-wise shuffling on those chromosomes, which does not cause any disruption when the overlap is non-cyclic.

Even with this modification, disruptions still occur when the overlapping is cyclic. Figure 7.12 shows a simple problem with four cyclically overlapping BBs. By shuffling from

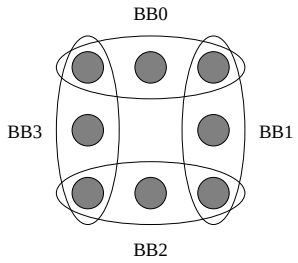


Figure 7.12: When the overlapping is cyclic, disruptions occur. By shuffling from BB_0 first, performing the population-wise BB shuffling on BB_1 and BB_3 does not disrupt these two BBs; however, BB_2 is disrupted.

BB_0 first, performing the population-wise BB shuffling on BB_1 and BB_3 does not disrupt these two BBs; however, BB_2 is disrupted.

When the overlap topology is cyclic, a certain number of BBs will be disrupted. The appropriate strategy is to preserve *stronger* BBs and disrupt *weaker* ones. Here, the strength of a BB is defined by how severe the disruption can be if the BB is crossed somewhere in the middle. As shown in Chapter 3.3, the loss in entropy serves as a decent indicator for the actual number of disruptions. Therefore, define the strength of a BB as:

$$Stength(\vec{x}) = \Sigma_i Entropy(x_i) - Entropy(\vec{x}), \tag{7.3}$$

where $\vec{x}$ is the BB, x_i is the i-th gene of the BB, and $Entropy(\vec{x})$ is the joint entropy of the BB.

Given the strength of every BB is calculated, the non-disruptive population-wise shuffling starts from the strongest BB. Among the BBs that overlaps with the first BB, the strongest is chosen as the next candidate. The procedure continues until the value of every gene is decided (Figure 7.13). Note that those BBs that cause a cycle are discarded. The crossover topology is like a tree where every node has only one single parent. In a 2D spin-glass problem with l spins and $m = 2l$ BBs, $(l-1)$ stronger BBs are preserved while $(l+1)$ weaker BBs are disrupted.

Figure 7.14 shows the scalability results of the `sequencing` recombination method with

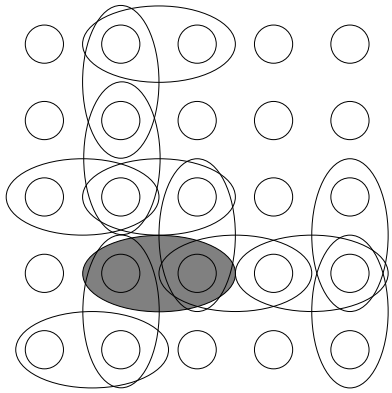


Figure 7.13: The `sequencing` recombination method working on a 5×5 2D spin-glass problem. The shadowed BB is the starting point (the *strongest* BB). The crossover sequencing topology is a tree structure.

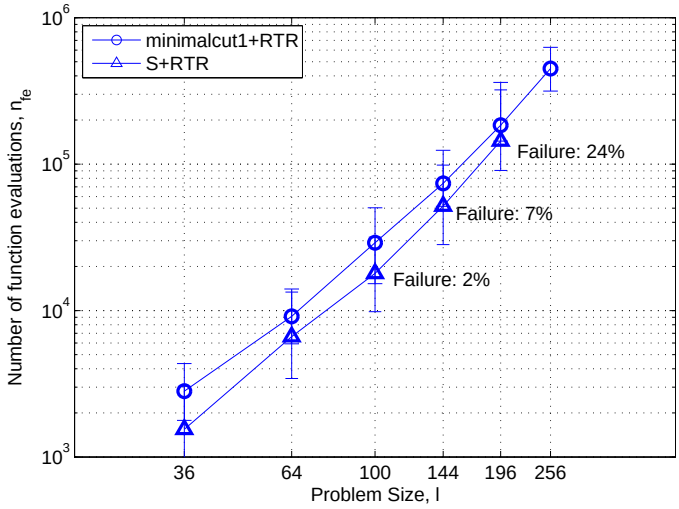


Figure 7.14: Scalability of `S+RTR` on the 2D spin-glass problem. `S+RTR` constantly uses fewer number of function evaluation than `minimalcut1+RTR`, but when the problem size becomes larger, more often the GA with `S+RTR` fails to converge within 10^8 function evaluations.

RTR (`S+RTR`) on the 2D spin-glass problems. For comparison, the result of `minimalcut1+RTR` is also plotted in the figure. From the figure, `S+RTR` constantly uses fewer number of function evaluations than `minimalcut1+RTR` does although the slope increases in a similar trend. One important thing to note is that when the problem size becomes larger, more often the GA with `S+RTR` could not find any global optima within 10^8 function evaluations.

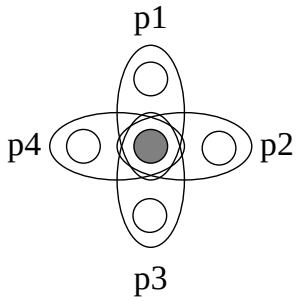


Figure 7.15: Well-informed decision for the 2D spin-glass problem. To decide the allele of gene x, all the information of p_1, p_2, p_3, and p_4 should be utilized. Ideally, the value of gene x can be determined by sampling $Pr(x|p_1, p_2, p_3, p_4)$.

7.6 Trial Four: Sequencing + Well-informed Decision + Niching

With only `sequencing`, problem occurs since about $\frac{m}{2}$ BBs are disrupted, even though disruptions only happen to weaker BBs. As indicated before, in the 2D spin-glass problem, every gene belongs to four different BBs. The recombination scheme in the previous section preserves at most two BBs and disrupts at least two others.

To prevent the recombination from disrupting too many BBs, the recombination needs to incorporate more information when deciding the value of the genes. Ideally, the recombination method should decide the allele of gene x_i by utilizing the information of all the genes of the BBs containing x_i. For instance, to decide the allele of gene x in Figure 7.15, all the information of p_1, p_2, p_3, and p_4 should be utilized. Implementation-wise, first calculate the conditional probability $Pr(x|p_1, p_2, p_3, p_4)$, then the allele of gene x is decided by sampling from the conditional probability.

A recombination method is constructed based on the above concept. The pseudo-code of the algorithm, called the `sequencing + well-informed decision` recombination algorithm (`S+W`), is shown in Algorithm 2. Note that not all the alleles of every genes is decided by all four other genes; otherwise, the sampling sequence is cyclic.

Figure 7.16 visually shows how `S+W` works on the 2D spin-glass problem. Suppose that

Algorithm 2 The `sequencing + well-informed decision (S+W)` recombination algorithm.

1. $S_{decided} = \phi$.
2. Randomly select a gene x from the strongest BB. Calculate $Pr(x)$ from the parent population. The value of gene x of chromosomes in the offspring population is decided by sampling $Pr(x)$. $S_{decided} \leftarrow S_{decided} \cup x$.
3. Select the strongest BB among all the BBs where some genes are in $S_{decided}$ and some other genes are not. $index = \text{argmax}_i Strength(\{BB_i | \exists x, x \in S_{decided}, x \in BB_i \text{ and } \exists y, y \notin S_{decided}, y \in BB_i\})$.
4. Randomly select a gene y from BB_{index}.
5. Create a set of parents of y. $\mathcal{P} = \{p_i | p_i \in S_{decided} \text{ and } \exists BB, p_i \in BB, y \in BB\}$.
6. Calculate the conditional probability $Pr(y|\mathcal{P})$ from the parent population. If the allele pattern of $\mathcal{P}$ does not exist in the parent population, use the marginal probability $Pr(y)$ instead.
7. The value of gene y of chromosomes in the offspring population is decided by sampling $Pr(y|\mathcal{P})$ or $Pr(y)$.
8. $S_{decided} \leftarrow S_{decided} \cup y$. Repeat steps 3 to 8 until all genes are in $S_{decided}$.

the BB containing genes x_{11} and x_{12} is the strongest, and that x_{11} is randomly selected. The values of gene x_{11} of the chromosomes in the offspring population are then sampled according to $p(x_{11})$, the proportion of x_{11} from the parent population. Among the four BBs that contains x_{11}, the BB containing x_{11} and x_{12} is the strongest by assumption, so next the algorithm decides the value of gene x_{12} by the proportion of x_{12} given the value of x_{11}, $p(x_{12}|x_{11})$. Now the values of genes x_{11} and x_{12} have been decided. Suppose that the strongest BB among the six BBs containing either x_{11} or x_{12} is the one containing gene x_7 and x_{12}. Then the value of x_7 is decided in the similar way. In Figure 7.16(d), the next step is to decide the value of gene x_6. Among those genes that share same BBs with x_6, two of them have their values decided (x_7 and x_{11}); therefore, the value of x_6 is decided according to $p(x_6|x_7, x_{11})$. The process continues until the values of all genes are decided.

Compared to `minimalcut1`, `minimalcut1+RTR`, and `S+RTR`, `S+W+RTR` shows a significant improvement in terms of number of function evaluations consumed (Figure 7.17). To obtain a more accurate comparison, `S+W+RTR` is performed on 1000 independent problem instances for each different problem size. Within 2×10^5 number of function evaluations, `S+W+RTR` is able to solve problems up to 20×20 spins.

Note that the slopes still increase with the problem size, while the increasing rate is slower than exponential (Figure 7.18). The followings discuss the possible reason and possible remedy. Review the crossover shown in Figure 7.11, there are different recombination probabilities between the BBs. For example, the probabilities that $BB_A = 1111$ recombines with $BB_B = \{1111, 1010, 1100, 0000, 0101, 0011\}$ can be calculated as $\left\{\frac{1}{3}, \frac{1}{3}, \frac{1}{3}, 0, 0, 0\right\}$, respectively. Similarly, the recombination probabilities for $BB_A = 1111$ and $BB_C = \{0000, 0101, 0000, 1111, 1010, 1111\}$) is $\left\{\frac{2}{9}, \frac{2}{9}, \frac{2}{9}, \frac{1}{9}, \frac{1}{9}, \frac{1}{9}\right\}$, respectively. The above calculations indicate that BB_C has a more uniformly distributed probability to recombine with BB_A than BB_B does. This observation is generally true for the `sequencing` recombination. In other words, the root has the most uniformly distributed recombination probability with the farthest leaf nodes. Recall that one of the keys for effective and efficient recombination

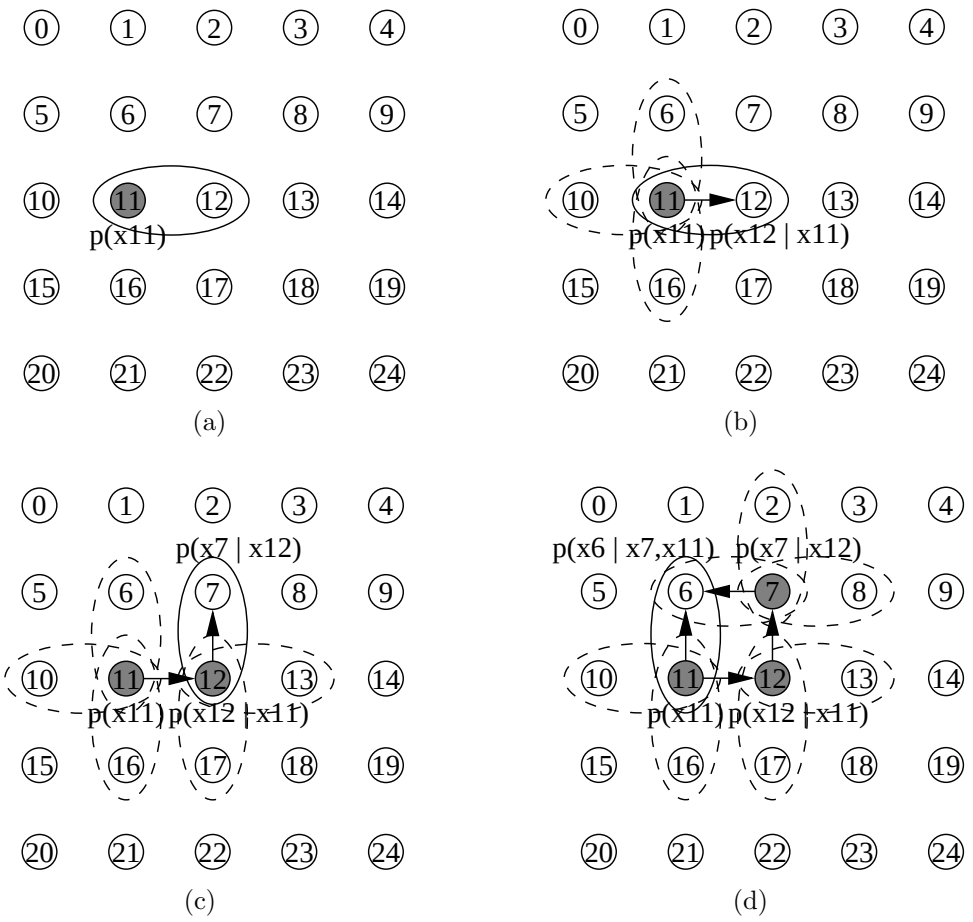


Figure 7.16: The `sequencing + well-informed decision (S+W)` algorithm performing on a 2D spin-glass problem.

is nondeterministic information exchange. Therefore, in order to have a more uniformly distributed recombination probability among BBs, a longer path in the sampling sequence is preferred. However, note that all the conditional probabilities are merely estimations. A longer path also amplifies estimation errors. A metric is needed to balance the tradeoff.

One simple way to introduce nondeterminism is to adopt the Boltzmann distribution to select the next BB to shuffle. A BB is selected with a probability of $ce^{-\frac{S}{T}}$, where c is a normalization constant, S is the strength of the BB, and T is a user-defined parameter. When $T \simeq 0$, the selection of the next BB will be uniformly random; a larger T emphasizes more on the strength. The main task would be to find an optimal T. However, the investigation

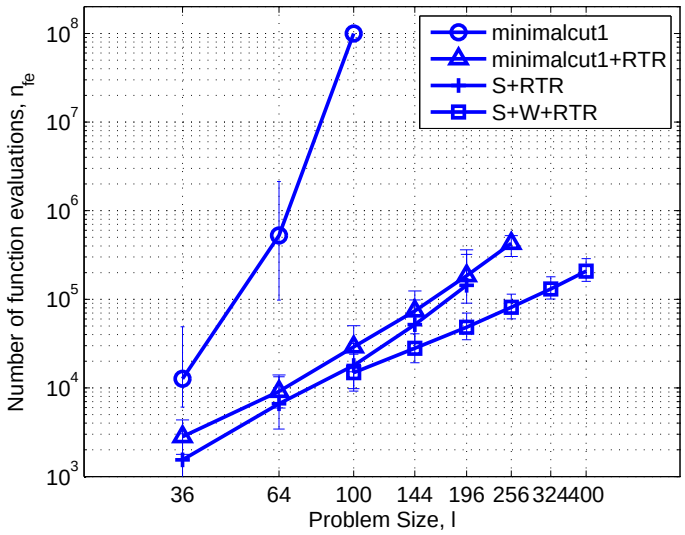


Figure 7.17: Comparison of `minimalcut`, `minimalcut+RTR`, `S+RTR`, and `S+W+RTR` on the 2D spin-glass problem. `S+W+RTR` outperforms other algorithms in terms of number of function evaluations consumed.

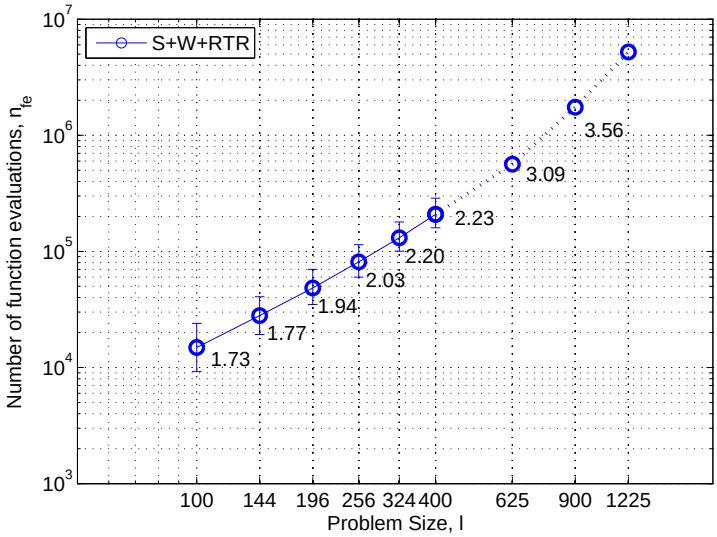


Figure 7.18: Scalability of the `sequencing + well-informed decision` algorithm on the 2D spin-glass problem with problem size up to 35×35. For problems with size larger than 20×20, the bisection run is performed on only one problem instance due to limited computational resource.

is beyond the scope of this thesis.

7.6.1 Discussion of results

From the series of experiments, three important keys to conquer overlap difficulty can be recognized:

1. Preservation of alternative solutions.

2. Proper sequencing.

3. Well-informed decision.

This thesis does not yet fully answer the question of how to achieve proper sequencing. Nevertheless, it acknowledges that a proper sequencing should respect the strengths of BBs and nondeterminism. If a recombination algorithm respects these key issues, it should scale well. This argument is well supported by the success of hBOA on the spin-glass problems (Pelikan & Goldberg, 2003; Pelikan, Ocenasek, Trebst, Troyer, & Alet, 2004).

Combined with local searchers, hBOA solves the spin-glass problem in polynomial time with an order that competes the current best algorithm ($\Theta(l^{3.5})$). However, the reason for hBOA's success was not comprehended. Experiments in this chapter demonstrate that hBOA's success relies on adopting RTR and Bayesian networks. RTR preserves alternative solutions, while Bayesian networks sample alleles in a proper sequence and make well-informed decisions.

7.7 Off-line Model Building: The Pros and the Cons

In the above experiments on the 2D spin-glass problem, the DSM analysis is performed off-line. This motivates the possibility of performing model building off-line. This section briefly discusses the pros and cons.

As indicated in Chapter 3.1, there are three basic types of interaction-detection methods: (1) statistical analysis/perturbation (2) interaction adaptation, and (3) probabilistic model

building. Of all three types, statistical analysis/perturbation methods are most suitable for off-line usage because they do not need promising individuals; instead, those methods favor a uniformly distributed population over the solution space, or perturbations from several individuals. The remaining of this section discusses the pros and cons to perform interaction-detection methods off-line.

Pros:

Saving Computational Time. When used off-line, the interaction-detection algorithm is performed only once, and speedup can be obtained. The speedup, if any, should be proportional to the number of generations of the GA.

To gain speedup, the off-line usage should not cause extra function evaluations for GAs. Chapter 4 has shown that to guarantee a high confidence of obtaining correct dependencies, roughly a population size of $O(l \log l)$ is needed, where l is the problem size. By assuming an infinite population size and perfect mixing, Mühlenbein and Schlierkamp-Voosen (1993) and Thierens and Goldberg (1994) gave the time-to-convergence model, which grows proportional to $O(\sqrt{l})$. The above models suggest a lower bound for the number of function evaluations $O(l^{1.5} \log l)$ for a GA with interaction-detection techniques.

If the interaction-detection techniques are performed off-line, and every BB is correctly identified, according to the gambler's ruin model (Harik, Cantú-Paz, Goldberg, & Miller, 1997), only $O(l^{0.5} \log l)$ population size is needed (the $\log l$ comes from the failure rate). Therefore, the lower bound for the number of function evaluations becomes $O(l \log l) + O(l^{0.5} \log l) \times O(\sqrt{l}) = O(l^{1.5} \log l)$, where the first $O(l \log l)$ comes from the off-line interaction detection, and $O(l^{0.5} \log l) \times O(\sqrt{l})$ comes from the production of the population size and the convergence time. This is surely just a loose lower bound because an infinite population size and perfect mixing is assumed in the time-

to-convergence model.

The above estimation suggests that off-line model building consumes the same order, if not less, of computational time as on-line model building. Since the model is built only once, there is a possibility to save the total computational time by some constant. However, when the interaction-detection method is applied off-line, a fewer number of individuals are investigated and the BB information might be less accurate. As a result, the inaccuracy of the BB information occurs at fixed positions and may cause the GA not to converge. This will be mentioned later when the disadvantages are discussed.

BB information reusability. Another reason to perform interaction detection off-line is that the BB information retrieved from one problem might be reusable for another problem. Take car designing as an example. Car designing usually has similar multiple objectives but the designs for different cars weight those objectives differently. For example, the design might emphasize on acceleration for one car while stress comfortability for another. However, no matter how different the design weights those objectives, some dependencies are invariant. For instance, one can imagine that the design of the brake system should depend more on the engine and less on the windshield wiper. In such cases, the BBs for the series of the problems are similar, and hence the BB information is reusable. After one of those problems is optimized by the GA with interaction-detection techniques, other problems could be optimized efficiently by a simple GA with the same BB information.

Population size estimation. Several population-sizing models have been proposed, including the decision-making model (Goldberg, Deb, & Clark, 1992) and the gambler's ruin model (Harik, Cantú-Paz, Goldberg, & Miller, 1997). To use those population-sizing models, several parameters are needed including the number of BBs m and the order of BBs k. The off-line usage of interaction-detection methods obtains those

parameters, and hence those models can be used to estimate the population size needed for the GA.

Cons:

Less interaction model error tolerance. In a worst-case scenario, a misidentified BB in the off-line BB information would make the BB difficult to converge correctly for GAs due to BB disruptions. When the interaction-detection method is performed for every generation, the disruption would not be that severe. If a BB is misidentified in some generation, as long as the interaction-detection method is not too inaccurate, the BB would be correctly identified in later generations with a high probability. Chapter 6.2 shows that as long as the number of misidentified BB e is less than $\Theta(\sqrt{m})$, where m is the number of BBs of the problem, the convergence of $(m-1)$ BBs is highly possible (Yu & Goldberg, 2004). However, if the interaction-detection method is performed off-line, and it misidentifies e BBs, in the worst case, only $(m-e)$ BBs would correctly converge.

BB structure invariance assumption. One of the assumptions for the off-line usage of interaction-detection method is that the BB structure is invariant. However, optimization problems can be dynamic and hierarchical. The BB structure in a dynamic problem varies with time, while the BB structure in a hierarchical problem varies with the degree of the GA convergence. In these cases, the interaction-detection method needs to be performed every several generations, and the scheme becomes a relaxed version of the on-line usage where interaction detection is performed every generation. When estimating the number of interaction model errors is possible, interaction detection can be performed only when the number of errors exceeds a predefined threshold.

7.8 Summary and Conclusions

This chapter investigates problems with overlap. It starts with a simple problem with cyclically overlapping BBs. A `minimalcut` algorithm is proposed and is shown to work well on the problem. However, when applied to the 2D spin-glass problem, where the overlapping is more complicated, `minimalcut` fails. From a series of experiments on the 2D spin-glass problem, this chapter recognizes three keys to solve problems with overlap:

1. Preservation of alternative solutions.

2. Proper sequencing.

3. Well-informed decision.

The off-line usage of DSM analysis is also demonstrated in this chapter. The pros and cons for performing model building off-line are discussed. If estimating the cost of model building and fitness function evaluation is possible, we can optimize the timing to perform model building.

Chapter 8

Future Work and Conclusions

This thesis presents a research project that develops advanced GA techniques for problems with modularity, hierarchy, and overlap. It first motivates the importance of problem decomposition for solving large-scale problems in complex systems. Interaction detection using mutual information metric emerges as the main mechanism for problem decomposition. Inspired by the organization theory, this thesis develops a DSM clustering technique and utilizes the technique to identify modularity for GAs. Combined with BB-wise crossover, DSMGA is constructed to solve boundedly difficult problems with modularity within subquadratic number of function evaluations. DSMGA is further extended with an explicit chucking scheme to solve problems via hierarchical decomposition. In preparation for overlap difficulty, several facetwise models are derived to understand the effect of interaction model errors on GA convergence. Through a series of experiments on the 2D spin-glass problem, this thesis recognizes keys to conquer overlap difficulty, and the arguments are empirically verified.

8.1 Future Work

Though much work can be done to further extend the work in this thesis, they fall into the following three major directions.

Extending the applicability. This thesis puts the three bare-bone types of interactions—modularity, hierarchy, and overlap, under the microscope and investigates them individually. However, real-world problems usually contain all three types

of interactions and maybe some other types of interactions not recognized in this thesis. In addition, the chromosomes in this thesis are binary encoded. Extending the work to real-valued domain will further enhance the applicability to many real-world optimizations.

Enhancing the efficiency. Existing efficiency enhancement techniques, including parallelization (Cantú-Paz, 2000), hybridization (Goldberg & Voessner, 1999; Sinha & Goldberg, 2003), time continuation (Goldberg, 1999; Srivastava & Golberg, 2001), and evaluation relaxation (Sastry, 2002) should be applicable to the work in this thesis with minor modification. More recently, a technique called *sporadic model building* (Pelikan, Sastry, & Goldberg, 2006) has been developed to alleviate the expensive computation time consumed for model building. In addition, some techniques have been developed to better utilize the explicit interaction model to achieve efficiency enhancement. These techniques will be discussed in greater detail later.

Developing additional theory. The possible extensions of the work discussed above need theoretical guidelines. In addition, plenty of work can be done to refine the theory developed in this thesis. Several interesting topics include analyzing the complexity of DSMGA+ and DSMGA++, defining the difficulty for problems with hierarchy and overlap, and designing a problem that deceives the model builder to understand the model-building difficulty.

To give a more concrete understanding of ideas in these three directions, the rest of this section discusses several interesting extensions and their challenges.

8.1.1 Toward the combination of modularity, hierarchy, and overlap

Modularity serves as the basis for hierarchy and overlap. Therefore, this thesis has investigated the cases for (modularity + hierarchy) and (modularity + overlap). The only task

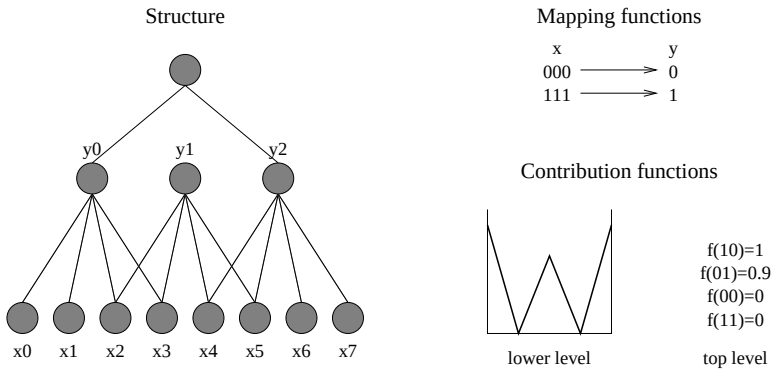


Figure 8.1: A challenging problem with hierarchy and overlap.

remaining is the combination of all three types of interactions.

Whether a problem is solvable or not by the work in this thesis depends on how the overlap and the hierarchy interact with each other. When overlap and hierarchy do not interact much, the methods described in this thesis should work well on the problem with no or minimal modifications. For instance, if overlaps only take place at the top level of the hierarchy, we can simply perform DSMGA+ to solve the problem on all lower levels and perform the recombination method for the overlap on the top level.

However, when overlap and hierarchy strongly interact with each other, the problem difficulty increases, making the problem extremely challenging. The following example gives a quick glance at how challenging the problem can be. In the example, the overlap increases the cardinality needed to solve the upper level of the hierarchy.

Figure 8.1 shows a problem with two levels of overlapping BBs. The contribution function of the lower level is a folded-trap function (Deb, Horn, & Goldberg, 1993) defined as:

$$f_l(u) = \begin{cases} 1 & \text{if } u = 0, 4 \\ 0.9 & \text{if } u = 2 \\ 0 & \text{if } u = 1, 3, \end{cases} \tag{8.1}$$

where u is the unitary of the four input genes. Let x denote the elements in a lower level

and y denote that in an upper level. The mapping from the lower level to the upper level is defined as

$$y = \begin{cases} 0 & \text{if } \vec{x} = 0000 \\ 1 & \text{if } \vec{x} = 1111 \\ - & \text{otherwise.} \end{cases} \tag{8.2}$$

The contribution function of the upper level is defined as

$$f_u(\vec{y}) = \begin{cases} 1 & \text{if } \vec{y} = 10 \\ 0.9 & \text{if } \vec{y} = 01 \\ 0 & \text{otherwise.} \end{cases} \tag{8.3}$$

Finally, the overall fitness value is defined as

$$fitness(\vec{x}) = f_l(x_0x_1x_2x_3) + f_l(x_2x_3x_4x_5) + f_l(x_4x_5x_6x_7) + w \cdot f_u(y_0y_2), \tag{8.4}$$

where w is a weighting coefficient.

The idea behind such a design is as follows. The lower level has two local optima, 0000 and 1111. Given that the three BBs overlap with each other, the optimal solutions at the lower level would be all 0's and all 1's. However, when the BBs at the lower level nearly converge to 0000 or 1111, the contribution from the upper level becomes stronger. At the lower level, 0000 and 1111 are indistinguishable in terms of the contribution to the fitness; however, the upper level encourages the first BB to be 1111 and the third BB to be 0000, breaking the second BB in the process. In other words, the lower level gives the highest contribution to the fitness when $\vec{x} = 00000000$ or 11111111, while the upper level gives the highest contribution to the fitness when $\vec{x} = 11110000$. The fitness values for these

chromosomes are:

$$fitness(00000000) = fitness(11111111) = 3. \tag{8.5}$$

$$fitness(11110000) = 2.9 + w. \tag{8.6}$$

Depending on the value of w, the global optima would be different. For $w > 0.1$, the optimal solution is to break the second BB. However, the structural chromosome compression scheme developed in Chapter 5 might have already compressed the second BB to its two most expressive schemata, 0000 and 1111, by the time the GA advances to the upper level and realizes that the best choice for the second BB is neither one of those two schemata.

The problem can be addressed in a more fundamental manner. Recall that one important key to conquer the hierarchical problem is to represent a BB in a lower level as a single variable in an upper level. This step is essential to prevent the problem difficulty from growing exponentially. However, because of overlap, the upper level is unsolvable without more information, which might have been lost due to the compact representation.

In the above example, preserving the schema 1100 for the second BB is required to solve the problem. Since the local expressiveness decides which schema to keep during the compression, 1111, 0000, 1010, 0110, 1001, 0101, and 0011 need to be kept in order to preserve 1100. That makes the cardinality for the upper level to be 8 even though the mapping function only suggests a cardinality of 2. In other words, the overlap increases the intrinsic hierarchical difficulty.

Two possible ways exist to handle this problem. One is to postpone the compression until the accurate decision can be made. The other is to adaptively choose the number of cardinality. Either way, accurate thresholds are needed to determine the timing for appropriate actions. A suitable test would be a larger-scale problem designed based on the concept of the above example, where the difficulty can be adjusted by varying the weights between the contribution from the lower and the upper levels.

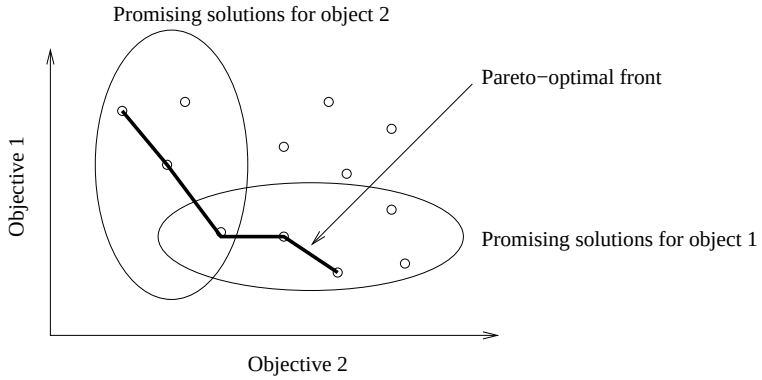


Figure 8.2: The Pareto-optimal front for a minimization problem with two objectives.

8.1.2 Multi-objective optimization

This thesis only considers problems with a single objective. However, many real-world problems have multiple objectives. The goal in multi-objective optimization is to find solutions that form the *Pareto-optimal front* (Figure 8.2). Solutions on the Pareto-optimal front excel any other solutions in the search space in at least one objective.

The major challenge for applying the problem decomposition concept to the multi-objective optimization is that the problem decomposition might be different for different objective. In this case, if interaction model is built blindly from the solutions of the current Pareto front, the model might not make sense at all.

One possible remedy is to build different models for each objective, and perform recombination by considering all models. For example, we can perform selection based on one particular objective and build a model from the selected solution candidates. The problem shown in Figure 8.3 contains 4 non-overlapping BBs for objective 1 and 3 non-overlapping BBs for objective 2. It is similar to a problem with 7 overlapping BBs. Therefore, the recombination method developed in Chapter 7 might be suitable for the multi-objective optimization.

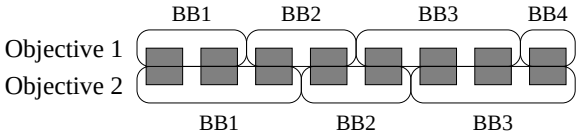


Figure 8.3: The recombination for problems with multi-objectives is similar to that for problems with overlap. The problem contains 4 BBs for objective 1 and 3 BBs for objective 2. It is similar to a problem with 7 overlapping BBs.

8.1.3 From binary to χ-ary and real-valued

Although this thesis focuses on binary chromosome encoding, the methodology should be easy to extend to the χ-ary domain in a straightforward manner. To extend the work to the real-valued domain, there are two possible directions: (1) discretizing the real values, and (2) handling real values directly. The methods in the first direction discretize the real-value domain into χ groups, and then treats the real-valued problem as a χ-ary problem. The methods in the second direction estimate the probabilistic distribution from the current instances and then sample new values from the distribution.

The major challenge for both directions is the finite sampling problem. The search space in real-valued domain is infinite, but the number of samples is finite. If the finite samples are not representative enough, the discretization can be skewed and the distribution estimated can be inaccurate. The new data points are then drifted away from the real optima. Therefore, population sizing is an important issue, and we will need some mechanism to ensure the data points are representative enough to reveal the fitness landscape.

8.1.4 Utilization of the explicit interaction model

As indicated in this thesis, the interaction model in DSMGA is explicit. Chapter 7 demonstrated how to take advantage of the explicitness by building the model off-line for small-scale problems and applying the model to larger-scale problems. Further, the explicit interaction model can also be used to develop the endogenous substructural fitness model (Sastry, Pelikan, & Goldberg, 2004), the substructural niching (Sastry, Abbass, & Goldberg, 2004), and

the on-line population-sizing adjustment scheme (Yu, Sastry, & Goldberg, 2005).

The idea in endogenous substructural fitness model is to reduce the number of real function evaluations by building a fitness surrogate based on the problem structure. The substructural niching utilizes the BB-wise distance measure. Ideally, it should preserve roughly the same number of promising subsolutions for each BB. The on-line population-sizing adjustment scheme utilizes the interaction model to retrieve necessary parameters for population-sizing models and adjusts the population size accordingly.

These developments are still in an early stage, and more investigations are needed to make them more applicable for real-world problems. Nevertheless, these researches show the value of the explicit interaction model, and more attention will be drawn to this direction.

8.2 Main Conclusions

Understanding problem structures and decomposing complex systems lay the foundation of this thesis. Compared to existing GAs, the work presented in this thesis distinguishes itself in two aspects: (1) it is capable of handling problems with modularity, hierarchy, and overlap and (2) it adopts explicit interaction models.

Among many other GAs, only hBOA is capable of handling problems with all three types of interactions. However, the interaction model in hBOA is implicit and opaque to users. On the contrary, the interaction model (DSM) in this thesis is explicit and transparent to users. In many real-world applications, the explicit knowledge of the problem structure is as valuable as finding a high-quality solution to the problem. Further, as presented in this thesis, the explicit interaction model can be used to develop advanced techniques.

The major contributions of this research come in two flavors. Technically, this thesis develops an automated dependency structure matrix clustering technique and utilizes it to design a competent black-box problem solver. These should benefit system analysis and design such as complex product and organization as well as optimization in many areas.

Scientifically, the facetwise models developed along the line of the DSMGA design help researchers better understand the relationship between model building errors and the resulting elongation in GA run duration. In addition, the explicit interaction model describes the dynamics of problem decomposition and helps researchers gain important insights through the transparency of the procedure. The work and knowledge presented in this thesis about problem decomposition should also help GA researchers design the next-generation problem-solver.

References

Abramowitz, M., & Stegun, L. (1970). *Handbook of mathematical functions*. New York: Dover.

Alexander, C. (1964). *Notes on the synthesis of form*. Boston, MA: Harvard Press.

Anderberg, M. (1973). *Cluster analysis for applications*. New York, NY: Academic Press.

Aporntewan, C., & Chongstitvatana, P. (2003). Building-block identification by simultaneity matrix. *Proceedings of the Genetic and Evolutionary Computation Conference*, 1566–1567.

Bäck, T. (1995). Generalized convergence models for tournament and (μ;λ) selection. *Proceedings of the Sixth International Conference on Genetic Algorithms (ICGA 1995)*, 2–8.

Baldwin, C., & Clark, K. (2000). *Design rules: The power of modularity*. Cambridge, MA: The MIT Press.

Barron, A., Rissenen, J., & Yu, B. (1998). The MDL principle in coding and modeling. *IEEE Transactions on Information Theory*, *44*(6), 2743–2760.

Blickle, T., & Thiele, L. (1995). A mathematical analysis of tournament selection. *Proceedings of the Sixth International Conference on Genetic Algorithms (ICGA'95)*, 9–16.

Broyden, C. (1965). A class of methods for solving nonlinear simultaneous equations. *Mathematics of Computation*, *19*(92), 577–5930.

Bui, T., & Moon, B. (1996). Genetic algorithm and graph partitioning. *IEEE Transactions on Computers*, *45*(7), 841–855.

Cantú-Paz, E. (2000). *Efficient and accurate parallel genetic algorithms*. Boston, MA: Kluwer.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2001). *Introduction to algorithms* (Second ed.). Cambridge, MA: The MIT Press.

Cover, T. M., & Thomas, J. A. (1991). *Elements of information theory* (pp. 18–26). New York: Wiley.

Deb, K., & Goldberg, D. E. (1993). Analyzing deception in trap functions. *Foundations of Genetic Algorithms 2*, 93–108.

Deb, K., Horn, J., & Goldberg, D. E. (1993). Multimodal deceptive functions. *Complex Systems*, *7*(2), 131–153.

Descartes, R. (1994). A discourse on the method of rightly conducting the reason, and seeking truth in the sciences [Veitch, J. (trans.)]. In Sorell, T. (Ed.), *A discourse on method: Meditations and principles* (pp. 3–57). London, UK: Everyman. Original work published 1637.

Eppinger, S. D. (2001). Innovation at the speed of information. *Harvard Business Review*, *79*(1), 149–158.

Eppinger, S. D., Whitney, D. E., Smith, R. P., & Gebala, D. A. (1994). A model-based method for organizing tasks in product development. *Research in Engineering Design*, *6*(1), 1–13.

Etxeberria, R., & Larrañaga, P. (1999). Global optimization using bayesian networks. *Proceedings of the Second Symposium on Artificial Intelligence Adaptive Systems*, 332–339.

Feller, W. (1966). *An introduction to probability theory*, Volume 2. New York: Wiley.

Fernandez, C. (1998). *Integration analysis of product architecture to support effective team co-location*. Master thesis, Massachusetts Institute of Technology.

Fischer, K. H., & Hertz, J. A. (1991). *Spin glasses*. New York, NY: Cambridge University Press.

Gaertler, M. (2002). *Clustering with spectral methods*. Master thesis, Universitat Konstanz, Germany.

Galluccio, A., & Loebi, M. (1999a). A theory of pfaffian orientations. I. perfect matchings and permunents. *Electronic Journal of Combinatorics*, *6*(1), Research Paper 6.

Galluccio, A., & Loebi, M. (1999b). A theory of pfaffian orientations. II. t-joins, k-cuts, and duality of enumeration. *Electronic Journal of Combinatorics*, *6*(1), Research Paper 7.

Garey, M., Johnson, D., & Stockmeyer, L. (1976). Some simplified NP-complete graph problems. *Theoretical Computer Science*, *1*(3), 237–267.

Glover, F. (1989). Tabu search–part I. *Operations Research Society of America (ORSA) Journal of Computing*, *1*(3), 190–206.

Glover, F. (1990). Tabu search–part II. *Operations Research Society of America (ORSA) Journal of Computing*, *2*(1), 4–32.

Goldberg, D. E. (1987). Simple genetic algorithms and the minimal, deceptive problem. In *Genetic Algorithms and Simulated Annealing* (Chapter 6, pp. 74–88). London: Pitman Publishing.

Goldberg, D. E. (1989a). Genetic algorithms and Walsh functions: Part II, deception and its analysis. *Complex Systems*, *3*, 153–171.

Goldberg, D. E. (1989b). *Genetic algorithms in search, optimization, and machine learning*. Reading, MA: Addison-Wesley.

Goldberg, D. E. (1989c). Sizing populations for serial and parallel genetic algorithms. *Proceedings of the Third International Conference on Genetic Algorithms*, 70–79.

Goldberg, D. E. (1999). Using time efficiently: Genetic-evolutionary algorithms and the continuation problem. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-1999)*, 212–219.

Goldberg, D. E. (2002). *The design of innovation: Lessons from and for competent genetic algorithms*. Boston, MA: Kluwer Academic Publishers.

Goldberg, D. E., Deb, K., & Clark, J. H. (1992). Genetic algorithms, noise, and the sizing of populations. *Complex Systems*, *6*, 333–362.

Goldberg, D. E., Deb, K., Kargupta, H., & Harik, G. (1993). Rapid, accurate optimization of difficult problems using fast messy genetic algorithms. *Proceedings of the Fifth International Conference on Genetic Algorithms*, 56–64.

Goldberg, D. E., Korb, B., & Deb, K. (1989). Messy genetic algorithms: Motivation, analysis, and first results. *Complex Systems*, *3*(5), 493–530.

Goldberg, D. E., Sastry, K., & Latoza, T. (2001a). On the supply of building blocks. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, 336–342.

Goldberg, D. E., Sastry, K., & Latoza, T. (2001b). On the supply of building blocks. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, 336–342.

Goldberg, D. E., & Voessner, S. (1999). Optimizing global-local search hybrids. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-1999)*, 220–228.

Gonzalez-Zugasti, J., Otto, K., & Baker, J. (2000). A method for architecting product platforms. *Research in Engineering Design*, *12*(2), 61–72.

Hansen, P., & Jaumard, B. (1997). Cluster analysis and mathematical programming. *Mathematical Programming*, *79*, 191–215.

Harik, G. (1999, February). *Linkage learning via probabilistic modeling in the ecga* (IlliGAL Report No. 99010). Urbana, IL: University of Illinois at Urbana-Champaign.

Harik, G., Cantú-Paz, E., Goldberg, D. E., & Miller, B. L. (1997). The gambler's ruin problem, genetic algorithms, and the sizing of populations. *Proceedings of the 1997 IEEE International Conference on Evolutionary Computation*, 7–12.

Harik, G. R. (1994). *Finding multiple solutions in problems of bounded difficulty* (IlliGAL Report No. 94002). Urbana, IL: University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory.

Harik, G. R. (1997). *Learning gene linkage to efficiently solve problems of bounded difficulty using genetic algorithms*. Doctoral dissertation, University of Michigan, Ann Arbor, ML.

Harik, G. R., Lobo, F. G., & Goldberg, D. E. (1998). The compact genetic algorithm. *Proceedings of IEEE International Conference on Evolutionary Computation*, 523–528.

Hartigan, J. (1975). *Clustering algorithms*. New York, NY: John Wiley and Sons.

Holland, J. H. (1975). *Adaptation in natural and artificial systems*. Ann Arbor, MI: University of Michigan Press.

Holland, J. H. (1995). *Hidden order how adaptation builds complexity*. Reading, MA: Perseus Books.

Hutter, M., & Zaffalon, M. (2005). Distribution of mutual information from complete and incomplete data. *Computational Statistics and Data Analysis*, *48*(3), 633–657.

Jian, A., Murty, M., & Flynn, P. (1999). Data clustering: A review. *ACM Computing Survey*, *31*(3), 264–323.

Kargupta, H. (1996). The performance of the gene expression messy genetic algorithm on real test functions. *Proceedings of 1996 IEEE International Conference on Evolutionary Computation*, 631–636.

Kirkpatric, S., Gelatt, C., & Vecchi, M. (1983). Optimization by simulated annealing. *Science*, *220*(4598), 671–680.

Kleiter, G. D. (1999). The posterior probability of bayes nets with strong dependences. *Soft Computing*, *3*, 162–173.

Kullback, S., & Leibler, R. A. (1951). On information and sufficiency. *Annual Mathematical Statistics*, *22*, 79–86.

Kusiak, A., & Huang, C. (1996). Development of modular products. *IEEE Transactions on Components, Packaging and Manufacturing Technology—Part A*, *19*(4), 523–538.

Larrañaga, P., & Lozano, J. (Eds.) (2002). *Estimation of distribution algorithms*. Boston, MA: Kluwer Academic Publishers.

Lutz, R. (2002). Recovering high-level structure of software systems using a minimum description length principle. *Proceedings of the 13th Irish International Conference on Artificial Intelligence and Cognitive Science (AICS)*, 61–69.

Martin, M., & Ishii, K. (2002). Design for variety: Developing standardized and modularized product platform architectures. *Research in Engineering Design*, *13*(4), 213–235.

McCord, K., & Eppinger, S. D. (1993). *Managing the integration problem in concurrent engineering* (Working Paper 3594). Cambridge, MA: MIT Sloan School of Management.

Meyer, M. H., & Lehnerd, A. P. (1997). *The power of product platforms*. New York, NY: The Free Press.

Miller, B. L. (1997). *Noise, sampling, and efficient genetic algorithms*. Doctoral dissertation, University of Illinois at Urbana-Champaign, Urbana, IL.

Miller, B. L., & Goldberg, D. E. (1995). Genetic algorithms, tournament selection, and the effects of noise. *Complex System*, *9*, 193–212.

Mitchell, M., Forrest, S., & Holland, J. H. (1992). The royal road for genetic algorithms: Fitness landscapes and GA performance. *Towards a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life*, 245–254.

Monien, B., & Sudborough, I. H. (1988). MIN-CUT is NP-complete for edge weighted trees. *Theoretical Computer Science*, *58*(1–3), 209–229.

Mühlenbein, H., Mahnig, T., & Rodriguez, A. O. (1999). Schemata, distributions and graphical models in evolutionary optimization. *Journal of Heuristics*, *5*, 215–247.

Mühlenbein, H., & Schlierkamp-Voosen, D. (1993). Predictive models for the breeder genetic algorithm: I. Continuous parameter optimization. *Evolutionary Computation*, *1*(1), 25–49.

Munetomo, M., & Goldberg, D. E. (1999). Identifying linkage groups by nonlinearity/nonmonotonicity detection. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-1999)*, *1*, 433–440.

Nascimento, H. A. D., & Eades, P. (2001). Interactive graph clustering based upon user hints. Paper presented at the Proceedings of the Second International Workshop on Soft Computing Applied to Software Engineering.

Naudts, B., & Naudts, J. (1998). The effect of spin-flip symmetry on the performance of the simple ga. *Parallel Problem Solving from Nature*, 67–76.

Pelikan, M. (2002). *Bayesian optimization algorithm: From single level to hierarchy.* Doctoral dissertation, University of Illinois at Urbana-Champaign.

Pelikan, M., & Goldberg, D. E. (2001). Escaping hierarchical traps with competent genetic algorithms. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, 511–518.

Pelikan, M., & Goldberg, D. E. (2003). Hierarchical BOA solves Ising spin glasses and MAXSAT. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2003)*, 1271–1282.

Pelikan, M., Goldberg, D. E., & Cantú-Paz, E. (1999). BOA: The Bayesian optimization algorithm. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-1999)*, *I*, 525–532.

Pelikan, M., & Mühlenbein, H. (1999). The bivariate marginal distribution algorithm. *Advances in Soft Computing—Engineering Design and Manufacturing*, 521–535.

Pelikan, M., Ocenasek, J., Trebst, S., Troyer, M., & Alet, F. (2004). Computational complexity and simulation of rare events of ising spin glasses. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2004)*, 36–47.

Pelikan, M., Sastry, K., & Goldberg, D. E. (2003). Scalability of the Bayesian optimization algorithm. *International Journal of Approximate Reasoning*, *31*(3), 221–258.

Pelikan, M., Sastry, K., & Goldberg, D. E. (2006). Sporadic model building for efficiency enhancement of hBOA. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2006)*, 405–412.

Pimmler, T., & Eppinger, S. D. (1994). Integration analysis of product decompositions. *Proceedings of the ASME International Conference on Design Theory and Methodology*, *DE-68*, 343–351.

Press, W. H., Teukolsky, S. A., Vetterling, W. T., & Flannery, B. P. (1992). *Numerical recipes in C: The art of scientific computing.* Cambridge, England: Cambridge University Press.

Rabani, Y., Rabinovich, Y., & Sinclair, A. (1998). A computational view of population genetics. *Random Structures and Algorithms*, *12*, 314–330.

Rattray, M., & Shapiro, J. L. (1997). Noisy fitness evaluations in genetic algorithms and the dynamics of learning. *Foundations of Genetic Algorithms*, *4*, 117–139.

Rechtin, E., & Maier, M. (1997). *The art of systems architecting.* Boca Raton, FL: CRC Press.

Reeves, C. (1993). Using genetic algorithms with small populations. *Proceedings of the Fifth International Conference on Genetic Algorithms*, 92–99.

Rissanen, J. (1978). Modeling by shortest data description. *Automatica*, *14*, 465–471.

Rissanen, J. (1999). Hypothesis selection and testing by the MDL principle. *Computer Journal*, *42*, 260–269.

Robertson, D., & Ulrich, K. (1998). Planning for product platforms. *Sloan Management Review*, *39*(4), 19–31.

Russell, S., & Norvig, P. (2003). Planning. In *Artificial Intelligence: A Modern Approach* (Second ed.). (Chapter 11–12, pp. 375–461). Upper Saddle River, NJ: Prentice Hall.

Santarelli, S., Yu, T.-L., Goldberg, D. E., Altshuler, E., O'Donnell, T., Southall, H., & Mailloux, R. (2006). Military antenna design using simple and competent genetic algorithms. *Mathematical and Computer Modelling*, *43*, 990–1022.

Sastry, K. (2002). *Evaluation-relaxation schemes for genetic and evolutionary algorithms.* Master thesis, University of Illinois at Urbana-Champaign, Urbana, IL.

Sastry, K., Abbass, H. A., & Goldberg, D. E. (2004). Sub-structural niching in non-stationary environments. *Proceedings of the Australian Artificial Intelligence Conference*, 873–885.

Sastry, K., & Goldberg, D. E. (2002). *Analysis of mixing in genetic algorithms: A survey* (IlliGAL Report No. 2002012). Urbana, IL: University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory.

Sastry, K., & Goldberg, D. E. (2004). Designing competent mutation operators via probabilistic model building of neighborhoods. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2004)*, *2*, 114–125.

Sastry, K., Pelikan, M., & Goldberg, D. E. (2004). Efficiency enhancement of genetic algorithms via building-block-wise fitness estimation. *Proceedings of the IEEE Conference on Evolutionary Computation*, 720–727.

Schloegel, K., Karypis, G., & Kumar, V. (1999). A new algorithm for multi-objective graph partitioning. *Proceedings of the European Conference on Parallel Processing (EuroPar 99)*, 322–331.

Shannon, C. E. (1948). A mathematical theory of communication. *The Bell System Technical Journal*, *27*, 379–423.

Sharman, D., & Yassine, A. (2004). Characterizing complex product architectures. *Systems Engineering Journal*, *7*(1), 35–60.

Sharman, D., Yassine, A., & Carlile, P. (2002). Architectural optimization using real options theory and dependency structure matrices. *Proceedings of the ASME 28th Design Automation Conference*, DAC–34119.

Simon, H. A. (1968). *The science of artifical*. Chambridge, MA: The MIT Press.

Simpson, T., Maier, J., & Mistree, F. (2001). Product platform design: Method and application. *Research in Engineering Design*, *13*(1), 2–22.

Sinha, A., & Goldberg, D. E. (2003, January). *A survey of hybrid genetic and evolutionary algorithms* (IlliGAL Report No. 2003004). Urbana, IL: University of Illinois at Urbana-Champaign.

Smith, J., & Fogarty, T. C. (1996). Recombination strategy adaptation via evolution of gene linkage. *Proceedings of the 1996 IEEE International Conference on Evolutionary Computation*, 826–831.

Srivastava, R., & Golberg, D. E. (2001). Verification of the theory of genetic and evolutionary continuation. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, 551–558.

Stirling, J. (1730). *Methodus differentialis, sive tractatus de summation et interpolation serierum infinitarium*. London. English translation by Holliday, J., *The Differential Method: A Treatise of the Summation and Interpolation of Infinite Series*, 1749.

Stone, R., Wood, K., & Crawford, R. (2000). A heuristic method for identifying modules for product architectures. *Design Studies*, *21*(1), 5–31.

Thebeau, R. (2001). *Knowledge management of system interfaces and interactions for product development processes*. Master thesis, Massachusetts Institute of Technology, Cambridge, MA.

Thierens, D., & Goldberg, D. E. (1993). Mixing in genetic algorithms. *Proceedings of the Fifth International Conference on Genetic Algorithms*, 38–45.

Thierens, D., & Goldberg, D. E. (1994). Convergence models of genetic algorithm selection schemes. *Parallel Problem Solving fron Nature (PPSN III)*, 119–129.

Toussaint, M. (2005). Compact genetic codes as a search strategy of evolutionary processes. *Foundations of Genetic Algorithms (FOGA 2005)*, 75–94.

Tsuji, M., Munetomo, M., & Akama, K. (2006). A crossover for complex building blocks overlapping. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2006)*, 1337–1344.

van Hoyweghen, C. (2001). Detecting spin-flip symmetry in optimization problems. In *Theoretical aspects of evolutionary computing* (pp. 423–437). New York, NY: Springer-Verlag.

van Hoyweghen, C., Goldberg, D. E., & Naudts, B. (2002). From TwoMax to the ising model: Easy and hard symmetrical problems. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2002)*, 626–633.

Wang, B., & Antonsson, E. (2004). Information measure for modularity in engineering design. *Proceedings of the ASME 2004 International Design Engineering Technical Conferences*, DETC–57515.

Watson, R. A., Hornby, G. S., & Pollack, J. B. (1998). Modeling building-block interdependency. *Parallel Problem Solving from Nature (PPSN-V)*, 97–106.

Watson, R. A., & Pollack, J. B. (1999). Hierarchically consistent test problems for genetic algorithms: Summary and additional results. *Late breaking papers at the Genetic and Evolutionary Computation Conference*, 292–297.

Watson, R. A., & Pollack, J. B. (2005). Modular interdependency in complex dynamical systems. *Artificial Life*, *11*(4), 445–457.

Whitfield, R., Smith, J., & Duffy, A. (2002). Identifying component modules. *Proceedings of the Seventh International Conference on Artificial Intelligence in Design AID02*, 571–592.

Widrow, B., & Hoff, M. E. (1960). Adaptive switching circuits. *IRE WESCON Convention Record*, *4*, 96–104.

Wright, A., Poli, R., Stephens, C., Landgon, W. B., & Pulavarty, S. (2004). An estimation of distribution algorithm based on maximum entropy. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2004)*, 343–354.

Yassine, A., Jogleker, N., Braha, D., Eppingher, S., & Whitney, D. (2003). Information hiding in product development: The design churn effect. *Research in Engineering Design*, *14*(3), 145–161.

Yu, T.-L., & Goldberg, D. E. (2004). Quality and efficiency of model building for genetic algorithms. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2000)*, 367–378.

Yu, T.-L., Goldberg, D. E., Yassine, A., & Chen, Y.-p. (2003). Genetic algorithm design inspired by organizational theory: Pilot study of a dependency structure matrix driven genetic algorithm. *Proceedings of Artificial Neural Networks in Engineering (ANNIE 2003)*, 327–332.

Yu, T.-L., Sastry, K., & Goldberg, D. E. (2005). Online population size adjusting using noise and substructural measurements. *Proceedings of the 2005 Congress on Evolutionary Computation Conference*, 2491–2498.

Yu, T.-L., Yassine, A., & Goldberg, D. E. (2005). *An information theoretic method for developing modular architectures using genetic algorithms* (IlliGAL Report No. 2005014). Urbana, IL: University of Illinois at Urbana-Champaign. Paper submitted to *Research in Engineering Design*.

Zakarian, A., & Rushton, G. (2001). Development of modular electrical systems. *IEEE Transactions on Mechatronic*, *6*(4), 507–520.

Author's Biography

Tian-Li Yu was born in Taipei, Taiwan on June 12, 1975. He graduated from the National Taiwan University in Taipei, Taiwan with a bachelor degree in Electrical Engineering in 1997. He arrived the University of Illinois at Urbana-Champaign to pursue graduate study in Computer Science in 2000 and became a member in the Illinois Genetic Algorithms Laboratory in 2001. He received his master degree from the University of Illinois at Urbana-Champaign in Computer Science in 2003. Following the completion of his Ph.D., Yu will engage in academic work in the National Taiwan University.